

AW882XX Android Driver(QCOM)

版本:

V3.3

时间:

2024 年 06 月

目录

1. 驱动说明.....	4
2. RX 驱动移植	4
2.1 AW882XX 驱动移植.....	4
2.1.1 DTS 配置.....	4
2.1.2 驱动配置.....	6
2.1.3 增加与 ADSP 通讯函数.....	6
2.1.4 BIN 文件配置.....	7
2.2 平台驱动配置.....	8
2.2.1 DAI_LINK 配置.....	8
2.3 平台通路配置.....	9
2.3.1 XML 配置.....	9
2.4 RX 驱动移植有效性验证	10
3. 算法集成.....	11
3.1 算法认证功能	11
4. TX 驱动移植.....	12
4.1 平台通路配置.....	12
4.1.1 TX 通路配置	12
4.1.2 TX CHANNEL 配置	13
4.1.3 FE DAI 配置.....	13
4.2 HAL 层移植.....	14
4.2.1 HAL 代码移植.....	14
4.2.2 XML 修改.....	17
4.3 TX 驱动移植有效性验证.....	18
5. 校准功能.....	19
5.1 校准目的	19
5.2 校准适配	19
5.2.1 校准文件保存路径适配.....	19
5.3 校准方式	20
5.3.1 CLASS 方式	20
5.3.2 MISC 方式	20
5.3.3 ATTR 方式.....	22
5.4 校准有效性验证.....	23
5.5 校准示例代码	23

6. 调试接口.....	24
6.1 设备节点	24
REG	24
RW	24
DRIVER_VER	24
DSP_RE	24
FADE_STEP	25
DBG_PROF	25
PHASE_SYNC	25
PRINT_DBG	25
ALGO_VER	25
MONITOR	26
MONITOR_UPDATE	26
6.2 KCONTROL 控件	26
7. 附录	26
7.1 关于校准	26
7.2 常见 TX_SWITCH 失效问题排查（APR 通信失败）	27
7.3 驱动 KO 编译配置	28
7.3.1 添加驱动文件	28
7.3.2 添加 ANDROID.MK 和 KBUILD	28
7.3.3 定义编译选项	32
7.3.4 添加编译选项	32
7.4 平台 I2C 总线动态变更	33
7.4.1 DTS 配置	33
7.4.2 DAI_LINK 配置	35

1. 驱动说明

驱动源文件	aw882xx_calib.c, aw882xx_calib.h, aw882xx_monitor.c, aw882xx_monitor.h, aw882xx.c, aw882xx.h, aw882xx_device.c, aw882xx_device.h, aw882xx_dsp.c, aw882xx_dsp.h, aw882xx_init.c, aw882xx_init.h, aw882xx_log.h, aw882xx_spin.c, aw882xx_spin.h, aw882xx_data_type.h, aw882xx_bin_parse.c, aw882xx_bin_parse.h
支持的产品（可以校准）	aw88257、aw88258、aw88261、aw88261s、aw88262、 aw88263、aw88263h、aw88263s、aw88264、aw88265、 aw88266、aw88266s、aw88270、aw88274、aw88299、 aw88461、aw88271
支持的产品（不可以校准）	aw88298、aw88298g、aw88266a、aw88230、aw88082、 aw88252（无需集成校准功能，请忽略 3/4/5 章的集成）
I ² C 地址范围	0x30/0x31/0x32/0x33/0x34/0x35/0x36/0x37

2. RX 驱动移植

2.1 AW882XX 驱动移植

2.1.1 DTS 配置

单 PA 配置方法

```
diff --git a/arch/arm/boot/dts/qcom/apq8096-dragonboard.dtsi
b/arch/arm/boot/dts/qcom/apq8096-dragonboard.dtsi
index f22db2e..a340a32 100644
--- a/arch/arm/boot/dts/qcom/apq8096-dragonboard.dtsi
+++ b/arch/arm/boot/dts/qcom/apq8096-dragonboard.dtsi
@@ -549,6 +549,8 @@
     i2c_x { /*x 表示对应的总线号*/
+
+        /* AWINIC AW882XX mono Smart PA */
+        aw882xx_smartpa@34 {
+            compatible = "awinic,aw882xx_smartpa";
+            reg = <0x34>;
+            reset-gpio = <&tlmm 84 0>; /*aw88230,aw88257,aw88261,aw88265,aw88082
不能配置*/
+            irq-gpio = <&tlmm 136 0x2008>;
+            sync-load = <1>; /*固件加载方式，qcom 平台默认使用同步加载*/
+            aw-tx-topo-id = <0x1000ff00>;
+            aw-rx-topo-id = <0x1000ff01>;
+            aw-tx-port-id = <0x1007>; /*根据客户实际所用 i2s tx 的 port id 进行配置*/
+            aw-rx-port-id = <0x1006>; /*根据客户实际所用 i2s rx 的 port id 进行配置*/

```

```
+          aw-re-min = <4000>;                /*Re 校准范围最小值 (mOhms) */
+          aw-re-max= <30000>;                /*Re 校准范围最大值 (mOhms) */
+          aw-cali-mode = "none";            /*带 IV 的产品无需配置此项*/
+          status = "okay";
+      };
+      /* AWINIC AW882XX mono Smart PA End */
+      /*Re 为阻抗值*/
```

多 PA 配置方法

```
diff --git a/arch/arm/boot/dts/qcom/apq8096-dragonboard.dtsi
b/arch/arm/boot/dts/qcom/apq8096-dragonboard.dtsi
index f22db2e..a340a32 100644
--- a/arch/arm/boot/dts/qcom/apq8096-dragonboard.dtsi
+++ b/arch/arm/boot/dts/qcom/apq8096-dragonboard.dtsi
@@ -549,6 +549,8 @@
     i2c_x {                /*x 表示对应的总线号*/
+
+        /* AWINIC AW882XX Smart PA */
+        aw882xx_smartpa@34 {
+            compatible = "awinic,aw882xx_smartpa";
+            reg = <0x34>;
+            reset-gpio = <&tlmm 84 0>;
+            irq-gpio = <&tlmm 136 0x2008>;
+            sync-load = <1>;
+            sound-channel = <0>; /*0:pri_l 1:pri_r 2:sec_l 3:sec_r*/
+            aw-tx-topo-id = <0x1000ff00>;
+            aw-rx-topo-id = <0x1000ff01>;
+            aw-tx-port-id = <0x1007>;
+            aw-rx-port-id = <0x1006>;
+            aw-re-min = <4000>;
+            aw-re-max= <30000>;
+            /*aw-cali-mode = "none";*/
+            status = "okay";
+        };
+        aw882xx_smartpa@35 {
+            compatible = "awinic,aw882xx_smartpa";
+            reg = <0x35>;
+            reset-gpio = <&tlmm 82 0>;
+            irq-gpio = <&tlmm 143 0x2008>;
+            sync-load = <1>;
+            sound-channel = <1>; /*0:pri_l 1:pri_r 2:sec_l 3:sec_r*/
+            aw-tx-topo-id = <0x1000ff00>;
+            aw-rx-topo-id = <0x1000ff01>;
+            aw-tx-port-id = <0x1007>;
+            aw-rx-port-id = <0x1006>;
+            aw-re-min = <4000>;
+            aw-re-max= <30000>;
+            /*aw-cali-mode = "none";*/
+            status = "okay";
+        };
+
+    /* AWINIC AW882XX Smart PA End */
```

2.1.2 驱动配置

驱动编译分为 ko 编译与内核编译两种方式，根据项目选择配置。

ko 编译配置

驱动 ko 编译配置见附录[驱动 KO 编译配置](#)。

内核编译配置

defconfig 编译配置选项

```
#add aw882xx smartpa
CONFIG_SND_SMARTPA_AW882XX=y
```

在内核创建 aw882xx 目录，添加 aw882xx 驱动文件

```
aw882xx_calib.c, aw882xx_calib.h, aw882xx_monitor.c,
aw882xx_monitor.h, aw882xx.c, aw882xx.h, aw882xx_device.c,
aw882xx_device.h, aw882xx_dsp.c, aw882xx_dsp.h, aw882xx_init.c,
aw882xx_init.h, aw882xx_log.h, aw882xx_spin.c, aw882xx_spin.h,
aw882xx_data_type.h, aw882xx_bin_parse.c, aw882xx_bin_parse.h
```

Kconfig 配置

```
config SND_SMARTPA_AW882XX
    tristate "SoC Audio for awinic aw882xxseries"
    depends on I2C
    help
        This option enables support for aw882xxseries Smart PA.
```

Makefile 配置

```
#for AWINIC AW882XX Smart PA
obj-$(CONFIG_SND_SMARTPA_AW882XX) += aw882xx/aw882xx.o
aw882xx/aw882xx_monitor.o aw882xx/aw882xx_init.o
aw882xx/aw882xx_dsp.o aw882xx/aw882xx_device.o
aw882xx/aw882xx_calib.o aw882xx/aw882xx_bin_parse.o
aw882xx/aw882xx_spin.o
```

2.1.3 增加与 ADSP 通讯函数

增加 afe 通信接口

请参照 patch 完成对 q6afe.c 的修改：

```
AW882XX_Driver_QCOM_v1.10.0 /*移植包名称 (以V1.10.0为例) */
├── ap
│   └── kernel
│       ├── 0001-Project-AW882XX_COPP.patch
│       ├── 0001-Project-AW882XX_COPP_V2.patch
│       ├── 0001-Project-AW882XX_driver.patch /*q6afe.c V3版本patch*/
│       └── 0001-Project-AW882XX_driver_params_v2.patch /*q6afe.c V2版本patch*/
```

增加 copp 控制接口（默认不合入）

请参照 patch 完成对 copp 控制接口的添加：

```
AW882XX_Driver_QCOM_v1.10.0 /*移植包名称 (以V1.10.0为例) */
├── ap
│   └── kernel
│       ├── 0001-Project-AW882XX_COPP.patch /*V3 版本copp接口添加patch*/
│       ├── 0001-Project-AW882XX_COPP_V2.patch /*V2 版本copp接口添加patch*/
│       ├── 0001-Project-AW882XX_driver.patch
│       └── 0001-Project-AW882XX_driver_params_v2.patch
```

2.1.4 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 += \
hardware/qcom/audio/configs/xxxx/aw882xx_acf.bin:$(TARGET_COPY_OUT_VENDOR)/firmware
/aw882xx_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"
};
```

(PS：调试阶段可直接 push bin 文件到/vendor/firmware/)

Bin 文件的选择

请根据平台信号输出格式，PA 数量以及产品名称在**移植包 config 目录下**选择 bin 文件。

2.2 平台驱动配置

2.2.1 DAI_LINK 配置

不同版本 linux kernel 的 dai_link 配置区分如下：

Kernel 5.4 之前的版本

1) 添加 awinic_codecs

单 PA 配置方法

```
diff --git a/sound/soc/msm/msm8996.c b/sound/soc/msm/msm8996.c
index ab04888..b5e4bf5 100644
--- a/sound/soc/msm/msm8996.c
+++ b/sound/soc/msm/msm8996.c
+struct snd_soc_dai_link_component awinic_codecs[] = {
+ {
+     .of_node = NULL,
+     .dai_name = "aw882xx-aif-6-34",
+     .name = "aw882xx_smartpa.6-0034",
+ },
+};
```

多 PA 配置方法

```
diff --git a/sound/soc/msm/msm8996.c b/sound/soc/msm/msm8996.c
index ab04888..b5e4bf5 100644
--- a/sound/soc/msm/msm8996.c
+++ b/sound/soc/msm/msm8996.c
+struct snd_soc_dai_link_component awinic_codecs[] = {
+ {
+     .of_node = NULL,
+     .dai_name = "aw882xx-aif-6-34",
+     .name = "aw882xx_smartpa.6-0034",
+ },
+ {
+     .of_node = NULL,
+     .dai_name = "aw882xx-aif-6-35",
+     .name = "aw882xx_smartpa.6-0035",
+ },
+};
```

2) 修改 DAI 接口

```
@@ -3775,8 +3777,10 @@ static struct snd_soc_dai_link
msm8996_common_be_dai_links[] = {
{
    .name = LPASS_BE_QUAT_MI2S_RX,
    .stream_name = "Quaternary MI2S Playback", /*以 Quaternary MI2S 接口举例*/
}
```



```
.cpu_dai_name = "msm-dai-q6-mi2s.3",
.platform_name = "msm-pcm-routing",
+#ifdef CONFIG_SND_SMARTPA_AW882XX
+ .num_codecs = ARRAY_SIZE(awinic_codecs),
+ .codecs = awinic_codecs,
+#else
. codec_name = "msm-stub-codec.1",
. codec_dai_name = "msm-stub-rx",
+#endif
. no_pcm = 1,
. dpcm_playback = 1,
. be_id = MSM_BACKEND_DAI_QUATERNARY_MI2S_RX,
. be_hw_params_fixup = msm_quat_mi2s_rx_be_hw_params_fixup,
. ops = &msm8996_quat_mi2s_be_ops,
. ignore_suspend = 1,
},
```

Kernel 5.4 及之后的版本

单 PA 配置方法

```
SND_SOC_DAILINK_DEFS(pri_mi2s_rx, /*以 pri_mi2s_rx 接口为例*/
DAILINK_COMP_ARRAY(COMP_CPU("msm-dai-q6-mi2s.0")),
/*以 I2C 总线为 0x6, 地址为 0x34 为例*/
DAILINK_COMP_ARRAY(COMP_CODEC("aw882xx_smartpa.6-0034", "aw882xx-aif-6-34")),
DAILINK_COMP_ARRAY(COMP_PLATFORM("msm-pcm-routing")));
```

多 PA 配置方法

```
SND_SOC_DAILINK_DEFS(pri_mi2s_rx, /*以 pri_mi2s_rx 接口为例*/
DAILINK_COMP_ARRAY(COMP_CPU("msm-dai-q6-mi2s.0")),
/*以双 PA 配置 I2C 总线为 0x6, 地址为 0x34, 0x35 为例*/
DAILINK_COMP_ARRAY(COMP_CODEC("aw882xx_smartpa.6-0034", "aw882xx-aif-6-34"),
COMP_CODEC("aw882xx_smartpa.6-0035", "aw882xx-aif-6-35")),
DAILINK_COMP_ARRAY(COMP_PLATFORM("msm-pcm-routing")));
```

2.3 平台通路配置

2.3.1 XML 配置

Mixer_paths_xxx.xml 文件修改

1) 增加初始化控件

```
@@ -334,6 +335,18 @@
<ctl name="SLIMBUS_0_RX Audio Mixer MultiMedia4" value="0" />
<ctl name="SLIMBUS_6_RX Port Mixer AUX_PCM_UL_TX" value="0" />
<ctl name="HDMI Mixer MultiMedia4" value="0" />
+ <!-- quat start -->
+ <!--以 QUAT_MI2S_RX 接口为例-->
+ <ctl name="QUAT_MI2S_RX Audio Mixer MultiMedia1" value="0" />
+ <ctl name="QUAT_MI2S_RX Audio Mixer MultiMedia2" value="0" />
+ <ctl name="QUAT_MI2S_RX Audio Mixer MultiMedia3" value="0" />
+ <ctl name="QUAT_MI2S_RX Audio Mixer MultiMedia4" value="0" />
+ <ctl name="QUAT_MI2S_RX Audio Mixer MultiMedia5" value="0" />
```

```
+ <ctl name="QUAT_MI2S_RX Audio Mixer MultiMedia6" value="0" />
+ <ctl name="QUAT_MI2S_RX Audio Mixer MultiMedia7" value="0" />
+ <ctl name="QUAT_MI2S_RX Audio Mixer MultiMedia8" value="0" />
+ <ctl name="QUAT_MI2S_RX Audio Mixer MultiMedia9" value="0" />
+ <ctl name="QUAT_MI2S_RX Audio Mixer MultiMedia10" value="0" />
<!-- HFP start -->
<ctl name="HFP_PRI_AUX_UL_HL Switch" value="0" />
<ctl name="SLIMBUS_0_RX Port Mixer AUX_PCM_UL_TX" value="0" />
```

2) 修改所有 usecase 的输出接口，以下以 deep-buffer-playback speaker 为例

```
@@ -608,7 +621,8 @@
    </path>

    <path name="deep-buffer-playback speaker">
-       <ctl name="SLIMBUS_0_RX Audio Mixer MultiMedia1" value="1" />
+       <!--ctl name="SLIMBUS_0_RX Audio Mixer MultiMedia1" value="1" /-->
+       <ctl name="QUAT_MI2S_RX Audio Mixer MultiMedia1" value="1" />
    </path>
```

3) 注释 path name="speaker"的内容

```
@@ -1884,7 +1901,7 @@
    </path>

    <path name="speaker">
-       <path name="speaker-lineout" />
+       <!--path name="speaker-lineout" /-->
    </path>
```

Audio_platform_info_xxx.xml 配置

所有使用的 device 都需要增加，以 SND_DEVICE_OUT_SPEAKER 为例

```
diff --git a/configs/msm8996/audio_platform_info.xml
b/configs/msm8996/audio_platform_info.xml
index e8fecec..22c40c2 100755
--- a/configs/msm8996/audio_platform_info.xml
+++ b/configs/msm8996/audio_platform_info.xml
@@ -47,6 +47,8 @@
    <param key="input_mic_max_count" value="4"/>
  </config_params>
  <backend_names>
+    <device name="SND_DEVICE_OUT_SPEAKER" interface="QUAT_MI2S_RX">
+      <device name="SND_DEVICE_OUT_HEADPHONES" backend="headphones"
interface="SLIMBUS_6_RX"/>
+      <device name="SND_DEVICE_OUT_LINE" backend="headphones"
interface="SLIMBUS_6_RX"/>
+      <device name="SND_DEVICE_OUT_ANC_HEADSET" backend="headphones"
interface="SLIMBUS_6_RX"/>
  </backend_names>

  /*以 QUAT_MI2S_RX 接口为例*/
```

2.4 RX 驱动移植有效性验证

按如上操作完成 RX 驱动移植，通过以下驱动 log 确认移植有效。

I2C 通信成功:

```
[Awinic] [6-0034] aw882xx_dai_drv_append_suffix: dai name [aw882xx-aif-6-34]
[Awinic] [6-0034] aw882xx_dai_drv_append_suffix: pstream_name name [Speaker_Playback-6-34]
[Awinic] [6-0034] aw882xx_dai_drv_append_suffix: cstream_name name [Speaker_Capture-6-34]
[Awinic] [6-0034] aw882xx_i2c_probe: dev_cnt 1
```

声卡注册成功:

```
[Awinic] [6-0034] aw882xx_codec_probe: enter
[Awinic] [6-0034] aw882xx_add_codec_controls: enter
[Awinic] [6-0034] aw882xx_request_firmware: load [aw882xx_acf.bin] , file size: [2016]
[Awinic] aw_dev_parse_check_acf_by_hdr: project name [A2113]
[Awinic] aw_dev_parse_check_acf_by_hdr: custom name [Awinic]
```

bin 文件加载成功:

```
[Awinic] [6-0034] aw_monitor_parse_vol_data_v_0_1_1: ==parse vol end ==
[Awinic] [6-0034] aw_dev_parse_skt_type: enter
[Awinic] [6-0034] aw_dev_parse_skt_type: get dsp data prof cnt is 0
[Awinic] [6-0034] aw_dev_parse_get_vaild_prof: get vaild profile:2
[Awinic] [6-0034] aw_dev_parse_acf: parse cfg success
[Awinic] [6-0034] aw_dev_soft_reset: soft reset done
[Awinic] [6-0034] aw_dev_reg_fw_update: amppd_st=0x0000
```

播放音乐, PA 发声:

```
[Awinic] [6-0034] aw_dev_set_intmask: done
[Awinic] [6-0034] aw_monitor_start: enter
[Awinic] [6-0034] aw_check_bop_status: enter
[Awinic] [6-0034] aw_check_bop_status: check done! bop status is 0
[Awinic] [6-0034] aw_device_start: done
[Awinic] [6-0034] aw882xx_start_pa: start success
```

3. 算法集成

请参考 Awinic 提供的算法集成文档进行集成。

3.1 算法认证功能

PA 工作时, 若出现规律播放白噪的情况, 时序如下: 播放音源 10s-->播放 3s 白噪-->播放音源 10s-->播放 3s 白噪-->.....

以上现象说明 awinic 算法认证失败, 请使能驱动中对应功能重新编译。

```
diff --git a/aw882xx_dsp.h b/aw882xx_dsp.h
index 998fa55..bce2a75 100644
--- a/aw882xx_dsp.h
+++ b/aw882xx_dsp.h
@@ -19,7 +19,7 @@

-/*#define AW_ALGO_AUTH_DSP*/
+#define AW_ALGO_AUTH_DSP

/*factor form 12bit(4096) to 1000*/
#define AW_DSP_RE_TO_SHOW RE(re) (((re) * (1000)) >> (12))
```

4. TX 驱动移植

4.1 平台通路配置

4.1.1 TX 通路配置

增加 TX 方向接口，以 QUAT_MI2S 为例进行添加

```
diff --git a/sound/soc/msm/qdsp6v2/msm-pcm-routing-v2.c
b/sound/soc/msm/qdsp6v2/msm-pcm-routing-v2.c
index ed7d4ed..1cabd90 100644
--- a/sound/soc/msm/qdsp6v2/msm-pcm-routing-v2.c
+++ b/sound/soc/msm/qdsp6v2/msm-pcm-routing-v2.c
@@ -10225,7 +10225,9 @@ static const char * const
slim0_rx_vi_fb_tx_rch_mux_text[] = {
    static const char * const mi2s_rx_vi_fb_tx_mux_text[] = {
        "ZERO", "SENARY_TX"
    };
-
+static const char * const quat_mi2s_rx_vi_fb_tx_mux_text[] = {
+    "ZERO", "QUAT_MI2S_TX"
+};
static const int const slim0_rx_vi_fb_tx_lch_value[] = {
    MSM_BACKEND_DAI_MAX, MSM_BACKEND_DAI_SLIMBUS_4_TX
};
@@ -10238,6 +10240,10 @@ static const int const mi2s_rx_vi_fb_tx_value[] = {
    MSM_BACKEND_DAI_MAX, MSM_BACKEND_DAI_SENARY_MI2S_TX
};

+static const int const quat_mi2s_rx_vi_fb_tx_value[] = {
+    MSM_BACKEND_DAI_MAX, MSM_BACKEND_DAI_QUATERNARY_MI2S_TX
+};
+
static const struct soc_enum slim0_rx_vi_fb_lch_mux_enum =
    SOC_VALUE_ENUM_DOUBLE(0, MSM_BACKEND_DAI_SLIMBUS_0_RX, 0, 0,
        ARRAY_SIZE(slim0_rx_vi_fb_tx_lch_mux_text),
@@ -10253,6 +10259,11 @@ static const struct soc_enum mi2s_rx_vi_fb_mux_enum =
    ARRAY_SIZE(mi2s_rx_vi_fb_tx_mux_text),
    mi2s_rx_vi_fb_tx_mux_text, mi2s_rx_vi_fb_tx_value);

+static const struct soc_enum quat_mi2s_rx_vi_fb_mux_enum =
+    SOC_VALUE_ENUM_DOUBLE(0, MSM_BACKEND_DAI_QUATERNARY_MI2S_RX, 0, 0,
+    ARRAY_SIZE(quat_mi2s_rx_vi_fb_tx_mux_text),
+    quat_mi2s_rx_vi_fb_tx_mux_text, quat_mi2s_rx_vi_fb_tx_value);
+
static const struct snd_kcontrol_new slim0_rx_vi_fb_lch_mux =
    SOC_DAPM_ENUM_EXT("SLIM0_RX_VI_FB_LCH_MUX",
        slim0_rx_vi_fb_lch_mux_enum, spkr_prot_get_vi_lch_port,
@@ -10268,6 +10279,11 @@ static const struct snd_kcontrol_new mi2s_rx_vi_fb_mux
    =
    mi2s_rx_vi_fb_mux_enum, spkr_prot_get_vi_lch_port,
    spkr_prot_put_vi_lch_port);

+static const struct snd_kcontrol_new quat_mi2s_rx_vi_fb_mux =
+    SOC_DAPM_ENUM_EXT("QUAT_MI2S_RX_VI_FB_MUX",
```

```
+ quat_mi2s_rx_vi_fb_mux_enum, spkr_prot_get_vi_lch_port,
+ spkr_prot_put_vi_lch_port);
+
static const struct snd_soc_dapm_widget msm_qdsp6_widgets[] = {
    /* Frontend AIF */
    /* Widget name equals to Front-End DAI name<Need confirmation>,
@@ -11242,7 +11258,8 @@ static const struct snd_soc_dapm_widget
msm_qdsp6_widgets[] = {
    &slim0_rx_vi_fb_rch_mux),
    SND_SOC_DAPM_MUX("PRI_MI2S_RX_VI_FB_MUX", SND_SOC_NOPM, 0, 0,
        &mi2s_rx_vi_fb_mux),
-
+ SND_SOC_DAPM_MUX("QUAT_MI2S_RX_VI_FB_MUX", SND_SOC_NOPM, 0, 0,
+     &quat_mi2s_rx_vi_fb_mux),
    SND_SOC_DAPM_MUX("VOC_EXT_EC_MUX", SND_SOC_NOPM, 0, 0,
        &voc_ext_ec_mux),
    SND_SOC_DAPM_MUX("AUDIO_REF_EC_UL1_MUX", SND_SOC_NOPM, 0, 0,
@@ -13450,9 +13467,11 @@ static const struct snd_soc_dapm_route intercon[] = {
    {"SLIM0_RX_VI_FB_LCH_MUX", "SLIM4_TX", "SLIMBUS_4_TX"},
    {"SLIM0_RX_VI_FB_RCH_MUX", "SLIM4_TX", "SLIMBUS_4_TX"},
    {"PRI_MI2S_RX_VI_FB_MUX", "SENARY_TX", "SENARY_TX"},
+ {"QUAT_MI2S_RX_VI_FB_MUX", "QUAT_MI2S_TX", "QUAT_MI2S_TX"},
    {"SLIMBUS_0_RX", NULL, "SLIM0_RX_VI_FB_LCH_MUX"},
    {"SLIMBUS_0_RX", NULL, "SLIM0_RX_VI_FB_RCH_MUX"},
    {"PRI_MI2S_RX", NULL, "PRI_MI2S_RX_VI_FB_MUX"},
+ {"QUAT_MI2S_RX", NULL, "QUAT_MI2S_RX_VI_FB_MUX"},
    {"PRI_TDM_TX_0", NULL, "BE_IN"},
    {"PRI_TDM_TX_1", NULL, "BE_IN"},
    {"PRI_TDM_TX_2", NULL, "BE_IN"},
```

4.1.2 TX channel 配置

配置 TX 的 channel 数为 2（单 PA 与双 PA 都需要配置为 2），以下示例作为参考。

```
diff --git a/kernel/msm-4.4/sound/soc/msm/sdm660-common.c b/kernel/msm-
4.4/sound/soc/msm/sdm660-common.c
index c30c42cf51..ae81c66530 100755
--- a/kernel/msm-4.4/sound/soc/msm/sdm660-common.c
+++ b/kernel/msm-4.4/sound/soc/msm/sdm660-common.c
@@ -236,7 +236,7 @@ static struct dev_config mi2s_rx_cfg[] = {

static struct dev_config mi2s_tx_cfg[] = {
    #ifdef CONFIG_SND_I2S_PRIMARY
-    [PRIM_MI2S] = {SAMPLING_RATE_48KHZ, SNDRV_PCM_FORMAT_S16_LE, 1},
+    [PRIM_MI2S] = {SAMPLING_RATE_48KHZ, SNDRV_PCM_FORMAT_S16_LE, 2},
    #else
    [PRIM_MI2S] = {SAMPLING_RATE_48KHZ, SNDRV_PCM_FORMAT_S16_LE, 1},
    #endif
```

4.1.3 FE DAI 配置

示例为 FE DAI 配置修改，可以作为参考

```
diff --git a/asoc/msm8996.c b/asoc/msm8996.c
index 71e01b9..9949b1f 100644
--- a/sound/soc/msm/msm8996.c
+++ b/sound/soc/msm/msm8996.c
```

```
@@ -2961,6 +3492,36 @@ static struct snd_soc_dai_link msm8996_common_dai_links[]
= {
    .codec_name = "snd-soc-dummy",
    .be_id = MSM_FRONTEND_DAI_VOICE2,
},
+ {
+     .name = "Quaternary MI2S RX Hostless",
+     .stream_name = "Quaternary MI2S RX Hostless Playback",
+     .cpu_dai_name = "QUAT_MI2S_RX_HOSTLESS",
+     .platform_name = "msm-pcm-hostless",
+     .dynamic = 1,
+     .dpcm_playback = 1,
+     .trigger = {SND_SOC_DPCM_TRIGGER_POST,
+         SND_SOC_DPCM_TRIGGER_POST},
+     .no_host_mode = SND_SOC_DAI_LINK_NO_HOST,
+     .ignore_suspend = 1,
+     .ignore_pmdown_time = 1,
+     .codec_dai_name = "snd-soc-dummy-dai",
+     .codec_name = "snd-soc-dummy",
+ },
+ {
+     .name = "Quaternary MI2S TX Hostless",
+     .stream_name = "Quaternary MI2S TX Hostless Capture",
+     .cpu_dai_name = "QUAT_MI2S_TX_HOSTLESS",
+     .platform_name = "msm-pcm-hostless",
+     .dynamic = 1,
+     .dpcm_capture = 1,
+     .trigger = {SND_SOC_DPCM_TRIGGER_POST,
+         SND_SOC_DPCM_TRIGGER_POST},
+     .no_host_mode = SND_SOC_DAI_LINK_NO_HOST,
+     .ignore_suspend = 1,
+     .ignore_pmdown_time = 1,
+     .codec_dai_name = "snd-soc-dummy-dai",
+     .codec_name = "snd-soc-dummy",
+ },
+ },
};
```

4.2 HAL 层移植

4.2.1 HAL 代码移植

1) 复制 aw882xx_feedback.c 文件到 audio_extn 目录中:



2) 修改 Android.mk 文件

```

diff --git a/hal/Android.mk b/hal/Android.mk
index 94c21bd..5a963fa 100644
--- a/hal/Android.mk
+++ b/hal/Android.mk
@@ -79,6 +79,10 @@ LOCAL_C_INCLUDES +=
$(TARGET_OUT_INTERMEDIATES)/KERNEL_OBJ/usr/include
LOCAL_C_INCLUDES +=
$(TARGET_OUT_INTERMEDIATES)/KERNEL_OBJ/usr/techpack/audio/include
LOCAL_ADDITIONAL_DEPENDENCIES += $(TARGET_OUT_INTERMEDIATES)/KERNEL_OBJ/usr

+#awinic add
+#LOCAL_CFLAGS += -DANDROID_R /*只有在 Android R 及以上版本才需要添加*/
+LOCAL_CFLAGS += -DAWINIC_SMARTPA_ENABLE
+LOCAL_SRC_FILES += audio_extn/aw882xx_feedback.c
+
ifeq ($(strip $(AUDIO_FEATURE_ENABLED_DLKM)),true)
LOCAL_HEADER_LIBRARIES += audio_kernel_headers
LOCAL_C_INCLUDES += $(TARGET_OUT_INTERMEDIATES)/vendor/qcom/opensource/audio-
kernel/include

```

3) 修改 audio_extn.h 文件，增加 Awinic 函数接口

```

diff --git a/hal/audio_extn/audio_extn.h b/hal/audio_extn/audio_extn.h
index d3e7a5f..6785c10 100644
--- a/hal/audio_extn/audio_extn.h
+++ b/hal/audio_extn/audio_extn.h
@@ -1106,4 +1105,13 @@ void audio_extn_ffv_append_ec_ref_dev_name(char *device_name);
#endif

+int audio_extn_aw882xx_start_feedback(struct audio_device *adev, snd_device_t
+snd_device);
+void audio_extn_aw882xx_stop_feedback(struct audio_device *adev, snd_device_t
+snd_device);
+

```

```
#endif /* AUDIO_EXTN_H */
```

4) 根据客户实际需求修改所需支持的设备类型

```
diff --git a/platform.c b/platform.c
index 1b6b5e0..7f48c4b 100644
--- a/platform.c
+++ b/platform.c
@@ -7506,16 +7506,25 @@ bool
platform_can_enable_spkr_prot_on_device(snd_device_t snd_device)
{
    bool ret = false;

+   ALOGV("%s [Awinic] %d\n", __func__, snd_device);
    if (snd_device == SND_DEVICE_OUT_SPEAKER ||
        snd_device == SND_DEVICE_OUT_SPEAKER_REVERSE ||
+       snd_device == SND_DEVICE_OUT_SPEAKER_WSA ||
        snd_device == SND_DEVICE_OUT_SPEAKER_VBAT ||
        snd_device == SND_DEVICE_OUT_VOICE_SPEAKER_VBAT ||
        snd_device == SND_DEVICE_OUT_VOICE_SPEAKER_2_VBAT ||
+       snd_device == SND_DEVICE_OUT_VOICE_SPEAKER_STEREO ||
        snd_device == SND_DEVICE_OUT_VOICE_SPEAKER ||
-       snd_device == SND_DEVICE_OUT_VOICE_SPEAKER_2) {
+       snd_device == SND_DEVICE_OUT_VOICE_SPEAKER_2 ||
+       snd_device == SND_DEVICE_OUT_VOICE_SPEAKER_WSA ||
+       snd_device == SND_DEVICE_OUT_VOICE_SPEAKER_2_WSA ||
+       snd_device == SND_DEVICE_OUT_VOIP_SPEAKER) {
        ret = true;
    }

+   if (ret) {
+       ALOGV("%s [Awinic] snd_device_out id %d is supported iv feedback\n",
+           __func__, snd_device);
+   } else {
+       ALOGV("%s [Awinic] unsupport snd_device_out id: %d\n",
+           __func__, snd_device);
+   }
    return ret;
}
```

5) 增加对 audio_extn_aw882xx_start_feedback 以及 audio_extn_aw882xx_stop_feedback 函数的调用

```
diff --git a/hal/audio_hw.c b/hal/audio_hw.c
index 3e3f72f..6574566 100644
--- a/hal/audio_hw.c
+++ b/hal/audio_hw.c
@@ -1096,7 +1096,8 @@ int enable_snd_device(struct audio_device *adev,
    }
    audio_extn_dev_arbi_acquire(snd_device);
    audio_route_apply_and_update_path(adev->audio_route, device_name);
-
}
```



```
+ /*Awinic Add*/
+ audio_extn_aw882xx_start_feedback( adev, snd_device );
    if (SND_DEVICE_OUT_HEADPHONES == snd_device &&
        !adev->native_playback_enabled &&
        audio_is_true_native_stream_active(adev)) {
@@ -1157,6 +1158,7 @@ int disable_snd_device(struct audio_device *adev,
        disable_snd_device(adev, new_snd_devices[i]);
    }
} else {
+     audio_extn_aw882xx_stop_feedback( adev, snd_device);
    audio_route_reset_and_update_path(adev->audio_route, device_name);
}
```

4.2.2 XML 修改

Mixer_paths_xxx.xml 文件修改

1) 在初始化列表中添加 IV 反馈的相关控制控件

```
@@ -179,6 +179,7 @@
    <ctl name="IIR1_INP2_MUX" value="ZERO" />
    <ctl name="SLIM0_RX_VI_FB_LCH_MUX" value="ZERO" />
    <ctl name="SLIM0_RX_VI_FB_RCH_MUX" value="ZERO" />
+   <ctl name="QUAT_MI2S_RX_VI_FB_MUX" value="ZERO" />
    <ctl name="VI_FEED_TX_Channels" value="Two" />
    <ctl name="AIF4_VI_Mixer_SPKR_VI_1" value="0" />
    <ctl name="AIF4_VI_Mixer_SPKR_VI_2" value="0" />
```

2) 修改 path name="spkr-vi-record"为 QUAT_MI2S 接口

```
@@ -1740,7 +1756,8 @@
    </path>

    <path name="spkr-vi-record">
-       <ctl name="SLIM0_RX_VI_FB_LCH_MUX" value="SLIM4_TX" />
+       <ctl name="QUAT_MI2S_RX_VI_FB_MUX" value="QUAT_MI2S_TX" />
+       <!--ctl name="SLIM0_RX_VI_FB_LCH_MUX" value="SLIM4_TX" /-->
    </path>
```

Audio_platform_info.xml 文件修改

1) 增加 QUAT_MI2S 配置

```
diff --git a/configs/msm8996/audio_platform_info.xml
b/configs/msm8996/audio_platform_info.xml
index e8fecec..22c40c2 100755
--- a/configs/msm8996/audio_platform_info.xml
+++ b/configs/msm8996/audio_platform_info.xml
@@ -47,6 +47,8 @@
    <param key="input_mic_max_count" value="4"/>
  </config_params>
  <backend_names>
    <device name="SND_DEVICE_OUT_SPEAKER" interface="QUAT_MI2S_RX">
```

```
+ <device name="SND_DEVICE_IN_CAPTURE_VI_FEEDBACK"
interface="QUAT_MI2S_TX">
  <device name="SND_DEVICE_OUT_HEADPHONES" backend="headphones"
interface="SLIMBUS_6_RX"/>
  <device name="SND_DEVICE_OUT_LINE" backend="headphones"
interface="SLIMBUS_6_RX"/>
  <device name="SND_DEVICE_OUT_ANC_HEADSET" backend="headphones"
interface="SLIMBUS_6_RX"/>
```

2) 如下图操作, 在 xml 中对 USECASE_AUDIO_SPKR_CALIB_TX 进行 pcm id 的配置

a. pcm id 确认方法如下:

```
msm8996:/ # cat proc/asound/pcm /*执行命令*/
00-00: MultiMedia1 (*) : : playback 1 : capture 1
00-01: MultiMedia2 (*) : : playback 1 : capture 1
00-02: VoiceMMModel (*) : : playback 1 : capture 1
00-03: VoIP (*) : : playback 1 : capture 1
00-04: MultiMedia3 (*) : : playback 1 : capture 1
00-05: SLIMBUS_0 Hostless (*) : : playback 1 : capture 1
00-06: Tertiary MI2S_TX Hostless Capture (*) : : capture 1
00-07: AFE-PROXY RX msm-stub-rx-7 : : playback 1
00-08: AFE-PROXY TX msm-stub-tx-8 : : capture 1
00-09: (Compress1) : : playback 1 : capture 1

00-38: (Compress8) : : playback 1
00-39: (Compress9) : : playback 1
00-40: CS-Voice (*) : : playback 1 : capture 1
00-41: Voice2 (*) : : playback 1 : capture 1
00-42: Quaternary MI2S_RX Hostless Playback (*) : : playback 1
00-43: Quaternary MI2S_TX Hostless Capture (*) : : capture 1 /*对应pcm id为43*/
00-44: SLIMBUS4 Capture tasha_vifeedback-44 : : capture 1
```

b. 配置 USECASE_AUDIO_SPKR_CALIB_TX

```
diff --git a/audio_platform_info.xml b/audio_platform_info.xml
index 8923f07..a284d73 100644
--- a/audio_platform_info.xml
+++ b/audio_platform_info.xml
@@ -34,6 +34,11 @@
    <device name="SND_DEVICE_IN_UNPROCESSED_QAUD_MIC" acdb_id="146"/>
    <device name="SND_DEVICE_IN_UNPROCESSED_HEADSET_MIC" acdb_id="147"/>
  </acdb_ids>
  <pcm_ids>
    <usecase name="USECASE_AUDIO_RECORD_LOW_LATENCY" type="in" id="19" />
+    <!--配置为实际对应的 pcm id-->
+    <usecase name="USECASE_AUDIO_SPKR_CALIB_TX" type="in" id="43" />
    <usecase name="USECASE_AUDIO_PLAYBACK_UCL" type="out" id="19" />
  </pcm_ids>
  <bit_width_configs>
    <device name="SND_DEVICE_OUT_SPEAKER" bit_width="24"/>
  </bit_width_configs>
```

4.3 TX 驱动移植有效性验证

播放音乐, locat 抓取日志, 确认配置是否正确:

E.g. Pcm tx start success

```
12-05 11:26:55.604 464 2290 D audio_hw_primary: enable_audio_route: apply
mixer and update path: spkr-vi-record
12-05 11:26:55.604 464 2290 D audio_route: Apply path: spkr-vi-record
12-05 11:26:55.605 1048 1584 W UsageStatsService: Event reported without a
package name
12-05 11:26:55.605 464 2290 D audio_hw_awinic_feedback:
audio_extn_aw882xx_start_feedback:[Awinic] the pcm id uc_info->id = 48, pcm_tx_id
= 43
12-05 11:26:55.631 2275 2275 I m.android.emai: The ClassLoaderContext is a
special shared library.
```

(PS: pcm_tx_id=43 对应的 stream_name = "Quaternary MI2S_TX Hostless Capture")

E.g. Pcm tx start failed

```
logcat
12-02 06:15:27.459 723 5039 D audio_hw_awinic_feedback:
audio_extn_aw882xx_start_feedback:[Awinic] the pcm id uc_info->id = 48, pcm_tx_id
= 43
12-02 06:15:27.472 723 5039 E audio_hw_awinic_feedback:
audio_extn_aw882xx_start_feedback:[Awinic] pcm start for TX failed
```

5. 校准功能

5.1 校准目的

针对喇叭保护需求,AW882XX 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保护算法中,达到喇叭保护的作用。

5.2 校准适配

5.2.1 校准文件保存路径适配

驱动 (aw882xx_calib.c) 中定义了保存校准值文件的路径如下:

```
#ifdef AW_CALI_STORE_EXAMPLE
/*write cali to persist file example*/
#define AWINIC_CALI_FILE "/mnt/vendor/persist/factory/audio/aw_cali.bin" /*保存校准值文件的路径*/
#define AW_INT_DEC_DIGIT 10
```

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 #
```

5.3 校准方式

awinic 提供了三种校准的方式，分别是 misc、class 和 attr 三种。

5.3.1 Class 方式

1) 节点功能

节点	功能
/sys/class/smarterpa/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/smarterpa/re25_calib	1.单独开启 re 校准 2.设置 re 值
/sys/class/smarterpa/f0_calib	单独开启 f0 校准
/sys/class/smarterpa/re_show	校准结束后获取 Re 值
/sys/class/smarterpa/f0_show	校准结束后获取 f0 值
/sys/class/smarterpa/re_range	获取校准 re 值有效范围

2) 校准步骤

- a. 播放静音文件；
- b. 校准 re:

```
msm8996:/sys/class/smarterpa #
msm8996:/sys/class/smarterpa # cat re25_calib
pri_l:6428 m0hms pri_r:6280 m0hms
msm8996:/sys/class/smarterpa #
```

c. 校准 f0:

```
msm8996:/sys/class/smarterpa #
msm8996:/sys/class/smarterpa # cat f0_calib
pri_l:832 pri_r:980
msm8996:/sys/class/smarterpa #
```

5.3.2 Misc 方式

AW882XX 驱动 misc 校准通过可执行文件来实现。按照以下步骤进行：

1) 获取可执行文件

```
AW882XX_Driver_QCOM_v1.10.0          /*移植包根目录 (以V1.10.0为例) */
├── ap
│   └── smartpa_cali
│       ├── aw882xx_cali_32          /*32位系统校准可执行文件*/
│       └── aw882xx_cali_64          /*64位系统校准可执行文件*/
│
│   └── example_source_code
│       ├── aw882xx_cali_attr_multi_mode.c
│       ├── aw882xx_cali_attr_single_dev_mode.c
│       └── aw882xx_cali_class_example.c
```

2) 配置可执行文件

```
C:\Users\AW882XX_Driver_QCOM_v1.10.0\ap\smartpa_cali>adb push aw882xx_cali_32 /system/bin/
aw882xx_cali_32: 1 file pushed. 0.6 MB/s (34504 bytes in 0.052s)

C:\Users\AW882XX_Driver_QCOM_v1.10.0\ap\smartpa_cali>adb shell chmod 0777 /system/bin/aw882xx_cali_32
```

3) 指令介绍

```
msm8996:/ # aw882xx_cali_32          /*执行命令*/
Calibration executables version: v0.3.10          /*文件版本号*/

./aw882xx_cali [dev_name] cmd [optional params]          /*命令格式*/

./aw882xx_cali [dev_name] cali [cali_re_time(ms)]          /*校准re、f0*/

./aw882xx_cali [dev_name] cali_re [cali_re_time(ms)]          /*校准re*/

./aw882xx_cali [dev_name] cali_f0 [noise]          /*校准f0*/

./aw882xx_cali [dev_name] get_spkr_status          /*获取实时re、te、f0*/

./aw882xx_cali [dev_name] get_spkr_st          /*获取实时re、te*/

./aw882xx_cali [dev_name] set_cali_re re_value1 [re_value2]          /*设置校准re值*/

./aw882xx_cali [dev_name] cali_f0_q [noise]          /*校准f0、q*/

./aw882xx_cali [dev_name] cali_all [cali_re_time(ms)]          /*校准re、f0、q*/

./aw882xx_cali [dev_name] get_re_range          /*获取校准re值有效范围*/

./aw882xx_cali dev_name set_params params_file          /*设置算法参数*/
```

参数解释（注：[]代表该选项可不填）

dev_name	用于校准单个设备，不同设备所用的 dev_name 与其在 dts 中配置的 sound-channel 相对应，其对应关系如下： 0: aw882xx_smartpa_l 1: aw882xx_smartpa_r 2: aw882xx_smartpa_sec_l 3: aw882xx_smartpa_sec_r 不填写时默认校准所有设备。
noise	填写 noise 参数用于校准 f0 时播放白噪； 若客户选择自行播放白噪，则无需填写该参数。
cali_re_time	用于设置校准 re 值的时间，单位（ms），最少不得小于 1000ms；

	不填写时将会使用默认校准时间 3000ms。
re_value1	需要设置到系统的校准电阻值（单位：mohm）。
params_file	算法参数对应的路径。

4) 校准步骤

- a. 播放静音文件；
- b. 启动校准：

```
msm8996:/ # aw882xx_cali_32 start_cali
[aw882xx_smartpa_l]cali_RE = 6429
[aw882xx_smartpa_r]cali_RE = 6346
[aw882xx_smartpa_l]cali_f0 = 836
[aw882xx_smartpa_r]cali_f0 = 979
```

5.3.3 Attr 方式

1) 节点路径

以 I2C 总线为 0x6，地址为 0x34 为例：

```
msm8996:/sys/bus/i2c/drivers/aw882xx_smartpa/6-0034 # /*节点路径*/
msm8996:/sys/bus/i2c/drivers/aw882xx_smartpa/6-0034 # ls
algo_ver cali_f0 dbg_prof dsp_re modalias name print_dbg reg uevent
awrw cali_re driver f0_show monitor phase_sync re_range rw
cali cali_time drv_ver fade_step monitor_update power re_show subsystem
```

2) 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	开启 re 和 f0 校准。
cali_re	1.单独开启 re 校准； 2.设置 re 值。
cali_f0	单独开启 f0 校准；
re_show	校准结束后获取 Re 值。
f0_show	校准结束后获取 f0 值。
re_range	获取校准 re 值有效范围

1) 校准步骤

- a. 播放静音文件；
- b. 校准 re:

```
msm8996:/sys/bus/i2c/drivers/aw882xx_smartpa/6-0034 #
msm8996:/sys/bus/i2c/drivers/aw882xx_smartpa/6-0034 # cat cali_re
pri_l:6371 mOhms pri_r:6380 mOhms
msm8996:/sys/bus/i2c/drivers/aw882xx_smartpa/6-0034 #
```

- c. 校准 f0:

```
msm8996:/sys/bus/i2c/drivers/aw882xx_smartpa/6-0034 #  
msm8996:/sys/bus/i2c/drivers/aw882xx_smartpa/6-0034 # cat cali_f0  
pri_l:830 pri_r:980  
msm8996:/sys/bus/i2c/drivers/aw882xx_smartpa/6-0034 #
```

5.4 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 方式校准两次为例:

```
msm8996:/sys/bus/i2c/drivers/aw882xx_smartpa/6-0034 # cat cali_re  
pri_l:6909 m0hms pri_r:6796 m0hms  
msm8996:/sys/bus/i2c/drivers/aw882xx_smartpa/6-0034 # cat cali_re  
pri_l:6855 m0hms pri_r:6802 m0hms  
msm8996:/sys/bus/i2c/drivers/aw882xx_smartpa/6-0034 #
```

2) 查看 re 值是否写入文件中:

```
msm8996:/ # cat mnt/vendor/persist/factory/audio/aw_cali.bin  
6855 6802  
msm8996:/ #  
msm8996:/ #
```

3) 播放音乐状态下, 查看 dsp_re 节点, 确认 dsp_re 节点 re 值与上述校准值相同:

```
msm8996:/ # cat sys/bus/i2c/drivers/aw882xx_smartpa/6-0034/dsp_re  
6854 /*从dsp中读取re值时需要经过定点化运算, 故存在误差, 误差值为1*/  
msm8996:/ # cat sys/bus/i2c/drivers/aw882xx_smartpa/6-0037/dsp_re  
6801 /*从dsp中读取re值时需要经过定点化运算, 故存在误差, 误差值为1*/  
msm8996:/ #
```

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。

5.5 校准示例代码

AW882XX 驱动提供了 attr、class 校准节点调用示例代码, 可供参考。

```
AW882XX_Driver_QCOM_v1.10.0 /*驱动移植包根目录 (以V1.10.0为例) */  
├── ap  
│   ├── smartpa_cali  
│   │   ├── aw882xx_cali_32  
│   │   └── aw882xx_cali_64  
│   └── example_source_code  
│       ├── aw882xx_cali_attr_multi_mode.c /*attr多设备同时校准方式示例代码*/  
│       ├── aw882xx_cali_attr_single_dev_mode.c /*attr单设备校准方式示例代码*/  
│       └── aw882xx_cali_class_example.c /*class校准方式示例代码*/
```


6. 调试接口

6.1 设备节点

AW882XX Driver 创建多个设备节点可供调试，路径是 sys/bus/i2c/drivers/aw882xx_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 aw882xx 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 aw882xx 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）

driver_ver

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

dsp_re

节点名字	dsp_re
功能描述	用于设置或者获取算法中设定的 re 值
使用方法	获取 re: cat dsp_re 设置 re: echo 7000 > dsp_re

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	场景切换 dbg 节点
使用方法	打开场景切换功能: echo 1 > dbg_prof 关闭场景切换功能: echo 0 > dbg_prof 获取节点状态: cat dbg_prof

phase_sync

节点名字	phase_sync
功能描述	相位同步功能开关节点
使用方法	打开相位同步功能: echo 1 > phase_sync 关闭相位同步功能: echo 0 > phase_sync 获取节点状态: cat phase_sync

print_dbg

节点名字	print_dbg
功能描述	i2c 写功能 dbg 打印节点
使用方法	i2c 写打印功能打开: echo 1 > print_dbg i2c 写打印功能关闭: echo 0 > print_dbg 获取节点状态: cat print_dbg

algo_ver

节点名字	algo_ver
功能描述	获取算法版本号
使用方法	获取算法版本号: cat algo_ver

monitor

节点名字	monitor
功能描述	用于开关 monitor 保护功能
使用方法	开启保护功能：echo 1 > monitor 关闭保护功能：echo 0 > monitor 保护模式获取：cat monitor

monitor_update

节点名字	monitor_update
功能描述	更新 monitor bin
使用方法	更新 monitor bin: echo 1 > monitor_update

6.2 Kcontrol 控件

Kcontrol	功能	实例
aw_dev_0_prof	模式选择	tinymix aw_dev_0_prof Music 选择 Music 模式 tinymix aw_dev_0_prof Receiver 选择 Receiver 模式
aw_dev_0_switch	开关芯片	tinymix aw_dev_0_switch Enable 开启芯片 tinymix aw_dev_0_switch Disable 关闭芯片
aw_dev_0_monitor	开关 monitor	tinymix aw_dev_0_monitor Enable 开启 monitor tinymix aw_dev_0_monitor Disable 关闭 monitor
aw882xx_rx_switch	开关 mec	tinymix aw882xx_rx_switch 0 关闭 mec tinymix aw882xx_rx_switch 1 开启 mec
aw882xx_tx_switch	开关 tx	tinymix aw882xx_tx_switch 0 关闭 tx tinymix aw882xx_tx_switch 1 开启 tx
aw882xx_spin_switch	设置旋转角度	tinymix aw882xx_spin_switch spin_90 旋转 90 度 tinymix aw882xx_spin_switch spin_180 旋转 180 度
aw882xx_fadein_us	设置淡入步进时间	tinymix aw882xx_fadein_us 500 设置淡入步进时间为 500
aw882xx_fadeout_us	设置淡出步进时间	tinymix aw882xx_fadeout_us 500 设置淡出步进时间为 500

7. 附录

7.1 关于校准

Awinic 提供了 misc ioctl 节点校准，device attr 节点校准以及 class 节点校准方式；

名称	说明
----	----

/dev/aw882xx_smartpa	支持多 PA 同时校准
/sys/class/smartpa	支持多 PA 同时校准
/sys/bus/i2c/drivers/aw882xx_smartpa/x-xx/	支持单个 PA 或者多个 PA 同时校准

代码默认 misc/class 多 PA 同时校准，device attr 节点既可以支持单个 PA 校准又可以支持多个 PA 校准；通过配置 aw882xx_calib.c 中的全局变量 is_single_cali 设置，FAE 或者客户可以根据需要设置：

如果只有单个 PA，所有节点均是单 PA 校准；

7.2 常见 tx_switch 失效问题排查（apr 通信失败）

如出现 tx_switch 控件失效，无法控制 TX 的问题。请检查驱动发送的 instance id（adsp 通讯版本 v3）与 acdb 中 tx 模块的 instance id 设置是否一致。

以下示例中驱动发送的 instance id 为 0x0，acdb 中 tx 模块的 instance id 为 0x8000。二者设置的 instance id 不一致，导致 kernel 向 dsp 发送消息后，dsp 无返回数据，apr 通信出错，最终导致 tx_switch 控件失效。

驱动中发送的 instance id 为 0x0：

```
int aw_send_afe_cal_apr(uint32_t param_id, void *buf, int cmd_size, bool write)
{
    int32_t result = 0, port_id = AFE_PORT_ID_AWDSP_RX;
    int32_t module_id = AFE_MODULE_ID_AWDSP_RX;
    uint32_t port_index = 0;
    uint32_t payload_size = 0;
    size_t len;
    struct rtac_cal_block_data *aw_cal = &(this_afe.aw_cal);
    struct mem_mapping_hdr mem_hdr;
    struct param_hdr_v3 param_hdr;

    pr_debug("%s: enter\n", __func__);

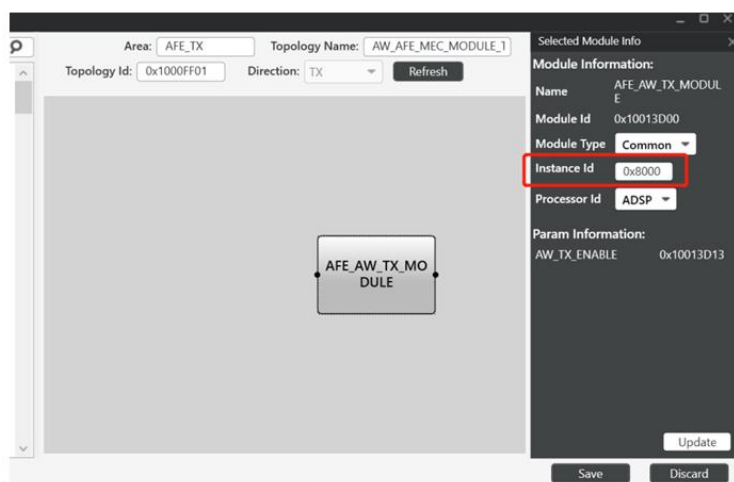
    if (param_id == AFE_PARAM_ID_AWDSP_TX_SET_ENABLE) {
        port_id = AFE_PORT_ID_AWDSP_TX;
        module_id = AFE_MODULE_ID_AWDSP_TX;
    }

    if (aw_cal->map_data.dma_buf == 0) {
        goto err;
    }

    /* Pack message header with data */
    param_hdr.module_id = module_id;
    param_hdr.instance_id = INSTANCE_ID_0;
    param_hdr.param_size = cmd_size;

    if (write) {
        param_hdr.param_id = param_id;
    }
}
```

acdb 中 tx 模块的 instance id 设置为 0x8000：



上述问题将 acdb 中 tx 模块的 instance id 改为 0x0000，apr 通信即可成功。

7.3 驱动 ko 编译配置

7.3.1 添加驱动文件

在 vendor/qcom/opensource/audio-kernel/asoc/codecs 路径下创建 aw882xx 文件夹，添加驱动源码。

7.3.2 添加 Android.mk 和 Kbuild

在 aw882xx 路径下创建 Android.mk 和 Kbuild 文件，以下基于 sdm450 平台配置举例，具体请参考平台其他模块 Android.mk 与 Kbuild。

Android.mk 配置

```
# Android makefile for audio kernel modules

# Assume no targets will be supported

AUDIO_CHIPSET := audio

# Build/Package only in case of supported target
ifeq ($(call is-board-platform-in-list,msm8953 msm8937 sdm710 qcs605),true)

LOCAL_PATH := $(call my-dir)

# This makefile is only for DLKM
ifneq ($(findstring vendor,$(LOCAL_PATH)),)

ifneq ($(findstring opensource,$(LOCAL_PATH)),)
    AUDIO_BLD_DIR := $(ANDROID_BUILD_TOP)/vendor/qcom/opensource/audio-kernel
```

```
endif # opensource

DLKM_DIR := $(TOP)/device/qcom/common/dlkm

# Build audio.ko as $(AUDIO_CHIPSET)_audio.ko
#####
# This is set once per LOCAL_PATH, not per (kernel) module
KBUILD_OPTIONS := AUDIO_ROOT=$(AUDIO_BLD_DIR)

# We are actually building audio.ko here, as per the
# requirement we are specifying <chipset>_audio.ko as LOCAL_MODULE.
# This means we need to rename the module to <chipset>_audio.ko
# after audio.ko is built.
KBUILD_OPTIONS += MODNAME=aw882xx_dlkm
KBUILD_OPTIONS += BOARD_PLATFORM=$(TARGET_BOARD_PLATFORM)
KBUILD_OPTIONS += $(AUDIO_SELECT)

#####
include $(CLEAR_VARS)
LOCAL_MODULE := $(AUDIO_CHIPSET)_aw882xx.ko
LOCAL_MODULE_KBUILD_NAME := aw882xx_dlkm.ko
LOCAL_MODULE_TAGS := optional
LOCAL_MODULE_DEBUG_ENABLE := true
LOCAL_MODULE_PATH := $(KERNEL_MODULES_OUT)
include $(DLKM_DIR)/AndroidKernelModule.mk
#####
endif # DLKM check
endif # supported target check
```

Kbuild 配置

```
# We can build either as part of a standalone Kernel build or as
# an external module. Determine which mechanism is being used
ifeq ($(MODNAME),)
    KERNEL_BUILD := 1
else
    KERNEL_BUILD := 0
endif

ifeq ($(KERNEL_BUILD), 1)
    # These are configurable via Kconfig for kernel-based builds
    # Need to explicitly configure for Android-based builds
    AUDIO_BLD_DIR := $(ANDROID_BUILD_TOP)/kernel/msm-4.9
    AUDIO_ROOT := $(AUDIO_BLD_DIR)/techpack/audio
```

```

endif

ifeq ($(KERNEL_BUILD), 0)
    ifeq ($(CONFIG_ARCH_SDM845), y)
        include $(AUDIO_ROOT)/config/sdm845auto.conf
        export
        INCS += -include $(AUDIO_ROOT)/config/sdm845autoconf.h
    endif
    ifeq ($(CONFIG_ARCH_SDM670), y)
        include $(AUDIO_ROOT)/config/sdm710auto.conf
        export
        INCS += -include $(AUDIO_ROOT)/config/sdm710autoconf.h
    endif
    ifeq ($(CONFIG_ARCH_SDM450), y)
        include $(AUDIO_ROOT)/config/sdm450auto.conf
        export
        INCS += -include $(AUDIO_ROOT)/config/sdm450autoconf.h
    endif
endif
endif

# As per target team, build is done as follows:
# Defconfig : build with default flags
# Slub      : defconfig + CONFIG_SLUB_DEBUG := y +
#            CONFIG_SLUB_DEBUG_ON := y + CONFIG_PAGE_POISONING := y
# Perf      : Using appropriate msmXXXX-perf_defconfig
#
# Shipment builds (user variants) should not have any debug feature
# enabled. This is identified using 'TARGET_BUILD_VARIANT'. Slub builds
# are identified using the CONFIG_SLUB_DEBUG_ON configuration. Since
# there is no other way to identify defconfig builds, QTI internal
# representation of perf builds (identified using the string 'perf'),
# is used to identify if the build is a slub or defconfig one. This
# way no critical debug feature will be enabled for perf and shipment
# builds. Other OEMs are also protected using the TARGET_BUILD_VARIANT
# config.

##### UAPI #####
UAPI_DIR := uapi
UAPI_INC := -I$(AUDIO_ROOT)/include/$(UAPI_DIR)

##### COMMON #####
COMMON_DIR := include
COMMON_INC := -I$(AUDIO_ROOT)/$(COMMON_DIR)

##### AW882XX #####

```

```
# for AW882XX PA
ifdef CONFIG_SND_SOC_AW882XX
    AW882XX_PA_OBJS += aw882xx.o //添加 aw882xx 相关驱动的所有.c 文件
endif

LINUX_INC += -include/linux

INCS += $(COMMON_INC) \
        $(UAPI_INC)

EXTRA_CFLAGS += $(INCS)

CDEFINES += -DANI_LITTLE_BYTE_ENDIAN \
            -DANI_LITTLE_BIT_ENDIAN \
            -DDOT11F_LITTLE_ENDIAN_HOST \
            -DANI_COMPILER_TYPE_GCC \
            -DANI_OS_TYPE_ANDROID=6 \
            -DPTT_SOCK_SVC_ENABLE \
            -Wall\
            -Werror\
            -D__linux__

KBUILD_CPPFLAGS += $(CDEFINES)

# Currently, for versions of gcc which support it, the kernel Makefile
# is disabling the maybe-uninitialized warning. Re-enable it for the
# AUDIO driver. Note that we must use EXTRA_CFLAGS here so that it
# will override the kernel settings.
ifeq ($(call cc-option-yn, -Wmaybe-uninitialized),y)
    EXTRA_CFLAGS += -Wmaybe-uninitialized
endif
#EXTRA_CFLAGS += -Wmissing-prototypes

ifeq ($(call cc-option-yn, -Wheader-guard),y)
    EXTRA_CFLAGS += -Wheader-guard
endif

ifeq ($(KERNEL_BUILD), 0)
    KBUILD_EXTRA_SYMBOLS += $(OUT)/obj/vendor/qcom/opensource/audio-
kernel/ipc/Module.symvers
    KBUILD_EXTRA_SYMBOLS += $(OUT)/obj/vendor/qcom/opensource/audio-
kernel/dsp/Module.symvers
    KBUILD_EXTRA_SYMBOLS += $(OUT)/obj/vendor/qcom/opensource/audio-
kernel/asoc/Module.symvers
```



```
KBUILD_EXTRA_SYMBOLS +=$(OUT)/obj/vendor/qcom/opensource/audio-
kernel/asoc/codecs/Module.symvers
KBUILD_EXTRA_SYMBOLS +=$(OUT)/obj/vendor/qcom/opensource/audio-
kernel/soc/Module.symvers
endif

# Module information used by KBuild framework

obj-$(CONFIG_SND_SOC_AW882XX) += aw882xx_dtlm.o
aw882xx_dtlm-y := $(AW882XX_PA_OBJS)

# inject some build related information
DEFINES += -DBUILD_TIMESTAMP=\"$(shell date -u +%Y-%m-%dT%H:%M:%SZ)\"
```

7.3.3 定义编译选项

```
--- a/config/sdm450auto.conf
+++ b/config/sdm450auto.conf
@@ -39,3 +39,4 @@ CONFIG_SND_SOC_ANALOG_CDC=m
CONFIG_SND_SOC_DIGITAL_CDC_LEGACY=m
CONFIG_SND_SOC_MSM_HDMI_CODEC_RX=m
CONFIG_WCD_DSP_GLINK=m
+++CONFIG_SND_SMARTPA_AW882XX=m

diff --git a/config/sdm450autoconf.h b/config/sdm450autoconf.h
index 1aca114..b01004e 100644
--- a/config/sdm450autoconf.h
+++ b/config/sdm450autoconf.h
@@ -55,3 +55,4 @@
#define CONFIG_SND_SOC_MSM_HDMI_CODEC_RX 1
#define CONFIG_COMMON_CLK 1
#define CONFIG_WCD_DSP_GLINK 1
+++#define CONFIG_SND_SMARTPA_AW882XX 1
```

7.3.4 添加编译选项

1) 在编译目录中添加新增的驱动目录

```
--- a/Android.mk
+++ b/Android.mk
@@ -13,6 +13,7 @@ $(shell rm -rf
$(PRODUCT_OUT)/obj/vendor/qcom/opensource/audio-
kernel/asoc/codecs/wcd934x/Module.symvers)
+$(shell rm -rf
$(PRODUCT_OUT)/obj/vendor/qcom/opensource/audiokernel/asoc/codecs/aw882xx/
```


Module.symvers)

```
@@ -22,6 +23,7 @@ include $(MY_LOCAL_PATH)/soc/Android.mk
include $(MY_LOCAL_PATH)/asoc/codecs/wcd934x/Android.mk
+include $(MY_LOCAL_PATH)/asoc/codecs/aw882xx/Android.mk
```

2) 在 device/qcom/XXX 目录下 (XXX 代表 ODM/OEM 的工程名) 添加相关配置:

BoardConfig.mk

```
BOARD_VENDOR_KERNEL_MODULES := $(KERNEL_MODULES_OUT)/audio_aw882xx.ko
```

XXX.mk

```
AUDIO_DLKM += audio_aw882xx.ko
```

init.target.rc 文件中添加编译出的 audio_aw882xx.ko 文件

```
/vendor/bin/modprobe -a -d /vendor/lib/modules audio_q6_pdr audio_q6_notifier
audio_snd_event audio_apr audio_adsp_loader audio_q6 audio_native audio_usf
audio_pinctrl_wcd audio_pinctrl_lpi audio_swr audio_platform audio_hdmi audio_stub
audio_wcd_core audio_wsa881x audio_bolero_cdc audio_wsa_macro audio_va_macro
audio_rx_macro audio_tx_macro audio_wcd938x audio_wcd938x_slave audio_machine_kona
audio_aw882xx
```

7.4 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。

7.4.1 DTS 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

```
diff --git a/arch/arm/boot/dts/qcom/apq8096-dragonboard.dtsi
b/arch/arm/boot/dts/qcom/apq8096-dragonboard.dtsi
index f22db2e..a340a32 100644
--- a/arch/arm/boot/dts/qcom/apq8096-dragonboard.dtsi
+++ b/arch/arm/boot/dts/qcom/apq8096-dragonboard.dtsi
@@ -549,6 +549,8 @@
     i2c_x { /*x 表示对应的总线号*/
+    /* AWINIC AW882XX mono Smart PA */
+    aw882xx_smartpa@34 {
+        compatible = "awinic,aw882xx_smartpa";
+        reg = <0x34>;
+        reset-gpio = <&tlmm 84 0>; /*aw88230,aw88257,aw88261,aw88265,aw88082
不能配置*/
+        irq-gpio = <&tlmm 136 0x2008>;
+        aw-tx-topo-id = <0x1000ff00>;
+        aw-rx-topo-id = <0x1000ff01>;
+        aw-tx-port-id = <0x1007>; /*根据客户实际所用 i2s tx 的 port id 进行配置*/
+        aw-rx-port-id = <0x1006>; /*根据客户实际所用 i2s rx 的 port id 进行配置*/
```

```
+         aw-re-min = <4000>;                /*Re 校准范围最小值 (mOhms) */
+         aw-re-max= <30000>;                /*Re 校准范围最大值 (mOhms) */
+         aw-cali-mode = "none";             /*带 IV 的产品无需配置此项*/
+         rename-flag = <1>;
+         status = "okay";
+     };
+     /* AWINIC AW882XX mono Smart PA End */
+     /*Re 为阻抗值*/
```

多 PA 配置方法

```
diff --git a/arch/arm/boot/dts/qcom/apq8096-dragonboard.dtsi
b/arch/arm/boot/dts/qcom/apq8096-dragonboard.dtsi
index f22db2e..a340a32 100644
--- a/arch/arm/boot/dts/qcom/apq8096-dragonboard.dtsi
+++ b/arch/arm/boot/dts/qcom/apq8096-dragonboard.dtsi
@@ -549,6 +549,8 @@
     i2c_x {          /*x 表示对应的总线号*/
+
+         /* AWINIC AW882XX Smart PA */
+         aw882xx_smartpa@34 {
+             compatible = "awinic,aw882xx_smartpa";
+             reg = <0x34>;
+             reset-gpio = <&tlmm 84 0>;
+             irq-gpio = <&tlmm 136 0x2008>;
+             sound-channel = <0>; /*0:pri_l 1:pri_r 2:sec_l 3:sec_r*/
+             aw-tx-topo-id = <0x1000ff00>;
+             aw-rx-topo-id = <0x1000ff01>;
+             aw-tx-port-id = <0x1007>;
+             aw-rx-port-id = <0x1006>;
+             aw-re-min = <4000>;
+             aw-re-max= <30000>;
+             /*aw-cali-mode = "none";*/
+             rename-flag = <1>;
+             status = "okay";
+         };
+         aw882xx_smartpa@35 {
+             compatible = "awinic,aw882xx_smartpa";
+             reg = <0x35>;
+             reset-gpio = <&tlmm 82 0>;
+             irq-gpio = <&tlmm 143 0x2008>;
+             sound-channel = <1>; /*0:pri_l 1:pri_r 2:sec_l 3:sec_r*/
+             aw-tx-topo-id = <0x1000ff00>;
+             aw-rx-topo-id = <0x1000ff01>;
+             aw-tx-port-id = <0x1007>;
+             aw-rx-port-id = <0x1006>;
+             aw-re-min = <4000>;
+             aw-re-max= <30000>;
+             /*aw-cali-mode = "none";*/
+             rename-flag = <1>;
+             status = "okay";
+         };
+     };
+     /* AWINIC AW882XX Smart PA End */
```

7.4.2 DAI_LINK 配置

codec_name 与 codec_dai_name 修改后缀为 sound-channel, 不同版本 linux kernel 的 dai_link 配置区分如下:

Kernel 5.4 之前的版本

3) 添加 awinic_codecs

单 PA 配置方法

```
diff --git a/sound/soc/msm/msm8996.c b/sound/soc/msm/msm8996.c
index ab04888..b5e4bf5 100644
--- a/sound/soc/msm/msm8996.c
+++ b/sound/soc/msm/msm8996.c
+struct snd_soc_dai_link_component awinic_codecs[] = {
+ {
+     .of_node = NULL,
+     .dai_name = "aw882xx-aif-0", /*修改后缀为 dts 节点对应的 sound-channel*/
+     .name = "aw882xx_smartpa_0",
+ },
+}
```

多 PA 配置方法

```
diff --git a/sound/soc/msm/msm8996.c b/sound/soc/msm/msm8996.c
index ab04888..b5e4bf5 100644
--- a/sound/soc/msm/msm8996.c
+++ b/sound/soc/msm/msm8996.c
+struct snd_soc_dai_link_component awinic_codecs[] = {
+ {
+     .of_node = NULL,
+     .dai_name = "aw882xx-aif-0", /*修改后缀为 dts 节点对应的 sound-channel*/
+     .name = "aw882xx_smartpa_0",
+ },
+ {
+     .of_node = NULL,
+     .dai_name = "aw882xx-aif-1", /*修改后缀为 dts 节点对应的 sound-channel*/
+     .name = "aw882xx_smartpa_1",
+ },
+}
```

4) 修改 DAI 接口

```
@@ -3775,8 +3777,10 @@ static struct snd_soc_dai_link
msm8996_common_be_dai_links[] = {
{
    .name = LPASS_BE_QUAT_MI2S_RX,
    .stream_name = "Quaternary MI2S Playback", /*以 Quaternary MI2S 接口举例*/
    .cpu_dai_name = "msm-dai-q6-mi2s.3",
    .platform_name = "msm-pcm-routing",
+ifdef CONFIG_SND_SMARTPA_AW882XX
+    .num_codecs = ARRAY_SIZE(awinic_codecs),
+    .codecs = awinic_codecs,
+else
    .codec_name = "msm-stub-codec.1",
    .codec_dai_name = "msm-stub-rx",
+endif
```

```
.no_pcm = 1,
.dpcm_playback = 1,
.be_id = MSM_BACKEND_DAI_QUATERNARY_MI2S_RX,
.be_hw_params_fixup = msm_quat_mi2s_rx_be_hw_params_fixup,
.ops = &msm8996_quat_mi2s_be_ops,
.ignore_suspend = 1,
},
```

Kernel 5.4 及之后的版本

单 PA 配置方法

```
SND_SOC_DAILINK_DEFS(pri_mi2s_rx, /*以 pri_mi2s_rx 接口为例*/
DAILINK_COMP_ARRAY(COMP_CPU("msm-dai-q6-mi2s.0")),
/*修改后缀为 dts 节点对应的 sound-channel*/
DAILINK_COMP_ARRAY(COMP_CODEC("aw882xx_smartpa_0", "aw882xx-aif-0")),
DAILINK_COMP_ARRAY(COMP_PLATFORM("msm-pcm-routing")));
```

多 PA 配置方法

```
SND_SOC_DAILINK_DEFS(pri_mi2s_rx, /*以 pri_mi2s_rx 接口为例*/
DAILINK_COMP_ARRAY(COMP_CPU("msm-dai-q6-mi2s.0")),
/*修改后缀为 dts 节点对应的 sound-channel*/
DAILINK_COMP_ARRAY(COMP_CODEC("aw882xx_smartpa_0", "aw882xx-aif-0")),
COMP_CODEC("aw882xx_smartpa_1", "aw882xx-aif-1")),
DAILINK_COMP_ARRAY(COMP_PLATFORM("msm-pcm-routing")));
```