

## AW882XX Android Driver(Qcom AudioReach)

版本: \_\_\_\_\_ V1.5

时间: \_\_\_\_\_ 2024 年 06 月

## 目录

1. 驱动说明.....	3
2. RX 驱动移植 .....	3
2.1 AW882XX 驱动移植.....	3
2.1.1 DTS 配置.....	3
2.1.2 驱动配置.....	4
2.1.3 定义编译选项.....	8
2.1.4 BIN 文件配置.....	8
2.2 平台驱动配置.....	9
2.2.1 DAI_LINK 配置.....	9
2.3 平台通路配置.....	10
2.3.1 XML 配置.....	10
2.4 驱动移植有效性验证.....	11
3. TX 驱动移植.....	12
3.1 HAL 层移植.....	12
3.2 XML 配置.....	12
4. 校准和低压保护功能 .....	13
5. 调试接口.....	13
5.1 设备节点 .....	13
REG .....	13
RW .....	13
DRIVER_VER .....	14
DSP_RE.....	14
FADE_STEP .....	14
DBG_PROF .....	14
PHASE_SYNC.....	14
PRINT_DBG.....	15
ALGO_VER.....	15
MONITOR .....	15
MONITOR_UPDATE.....	15
5.2 KCONTROL 控件.....	15

1. 驱动说明

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 章的集成）
I <sup>2</sup> 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

打开/vendor/qcom/proprietary/audio-devicetree/xxxx.dtsi 文件并进行 DTS 配置

单 PA 配置方法

```

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,aw88261,aw88265,aw88257,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-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

```

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-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-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

### 添加源文件

在 `/vendor/qcom/opensource/audio-kernel/asoc/codecs` 路径下创建 `aw882xx` 文件夹并添加驱动源文件

```
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
```

### 开启 AudioReach 宏

在 `aw882xx_dsp.h` 文件中注释 `AW_QCOM_PLATFORM` 宏, 打开 `AW_AUDIOREACH_PLATFORM` 宏

```
#ifndef __AW882XX_DSP_H__
#define __AW882XX_DSP_H__

/*#define AW_QCOM_PLATFORM*/
#define AW_AUDIOREACH_PLATFORM
```

### 添加 Makefile 和 Kbuild

在 `aw882xx` 路径下创建 `Makefile` 和 `Kbuild` 文件, 以下基于 `SM8450` 平台, **WAIPIO** 声卡配置举例, 具体请参考平台其他模块 `Makefile` 与 `Kbuild`。

#### 1) 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 := $(shell pwd)/kernel/msm-5.4
    AUDIO_ROOT := $(AUDIO_BLD_DIR)/techpack/audio
endif

ifeq ($(KERNEL_BUILD), 0)
    ifeq ($(CONFIG_ARCH_LAHAINA), y)
        include $(AUDIO_ROOT)/config/lahainaauto.conf
        INCS += -include $(AUDIO_ROOT)/config/lahainaautoconf.h
    endif
    ifeq ($(CONFIG_ARCH_WAPIO), y)
        include $(AUDIO_ROOT)/config/wapioauto.conf
        INCS += -include $(AUDIO_ROOT)/config/wapioautoconf.h
    endif
    ifeq ($(CONFIG_ARCH_LITO), y)
        include $(AUDIO_ROOT)/config/litoauto.conf
        INCS += -include $(AUDIO_ROOT)/config/litoautoconf.h
    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/audio
UAPI_INC := -I$(AUDIO_ROOT)/include/$(UAPI_DIR)

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

##### AWINIC #####

# for AW882XX Codec
ifdef CONFIG_SND_SOC_AW882XX
    AW882XX_OBJS += aw882xx.o
    AW882XX_OBJS += aw882xx_device.o
    AW882XX_OBJS += aw882xx_calib.o
    AW882XX_OBJS += aw882xx_monitor.o
    AW882XX_OBJS += aw882xx_init.o
    AW882XX_OBJS += aw882xx_dsp.o
    AW882XX_OBJS += aw882xx_bin_parse.o
    AW882XX_OBJS += aw882xx_spin.o
endif

LINUX_INC += -Iinclude/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

# Module information used by KBuild framework
obj-$(CONFIG_SND_SOC_AW882XX) += aw882xx_dlm.o
aw882xx_dlm-y := $(AW882XX_OBJS)

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

## 2) Makefile 配置

Makefile 编写参考如下，请根据平台决定是否添加

```
modules:
$(MAKE) -C $(KERNEL_SRC) M=$(M) modules $(KBUILD_OPTIONS)
VERBOSE=1
modules_install:
$(MAKE) M=$(M) -C $(KERNEL_SRC) modules_install
clean:
$(MAKE) -C $(KERNEL_SRC) M=$(M) clean
```

## 3) Kbuild 文件修改

修改 [/vendor/qcom/opensource/audio-kernel/asoc/codecs](#) 路径下的 Kbuild.

```
diff --git a/asoc/codecs/Kbuild b/asoc/codecs/Kbuild
index 545904f..f257823 100644
--- a/asoc/codecs/Kbuild
+++ b/asoc/codecs/Kbuild
@@ -260,6 +260,7 @@ ifeq ($(KERNEL_BUILD), 1)
obj-y += lpas-cdc/
obj-y += wsa883x/
obj-y += rouleaur/
+ obj-y += aw882xx/
endif
# Module information used by KBuild framework
obj-$(CONFIG_WCD9XXX_CODEC_CORE) += wcd_core_dlm.o
```

## 4) Android.mk 文件修改

修改 [/vendor/qcom/opensource/audio-kernel](#) 路径下的 Android.mk.

```
diff --git a/Android.mk b/Android.mk
index a3bb903..021e820 100644
```

```
--- a/Android.mk
+++ b/Android.mk
@@ -374,6 +374,15 @@ include
$(DLKM_DIR)/Build_external_kernelmodule.mk
#####
+include $(CLEAR_VARS)
+LOCAL_SRC_FILES := $(AUDIO_SRC_FILES)
+LOCAL_MODULE := aw882xx_dlk.ko
+LOCAL_MODULE_KBUILD_NAME := asoc/codecs/aw882xx/aw882xx_dlk.ko
+LOCAL_MODULE_TAGS := optional
+LOCAL_MODULE_DEBUG_ENABLE := true
+LOCAL_MODULE_PATH := $(KERNEL_MODULES_OUT)
+include $(DLKM_DIR)/Build_external_kernelmodule.mk
```

### 2.1.3 定义编译选项

- 1) 在/vendor/qcom/opensource/audio-kernel/config/waipioauto.conf 文件中添加配置

```
export CONFIG_SND_SOC_AW882XX=m
```

- 2) 在/vendor/qcom/opensource/audio-kernel/config/waipioautoconf.h 文件中添加配置

```
#define CONFIG_SND_SOC_AW882XX 1
```

### 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：

#### 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

根据平台 I2S 输出格式以及 PA 数量选择 bin 文件，以 aw88230 为例：





## 2.2 平台驱动配置

### 2.2.1 DAI\_LINK 配置

1) 在 `msm_dailink.h` 文件中，添加 DAI\_LINK 的配置，以 `quin_mi2s` 为例。

#### 单 PA 配置方法

```

#ifdef CONFIG_SND_SOC_AW882XX
SND_SOC_DAILINK_DEFS(quin_mi2s_rx_smartpa,
    DAILINK_COMP_ARRAY(COMP_CPU("snd-soc-dummy-dai")),
    /*以 I2C 总线为 0x0，地址为 0x34 为例*/
    DAILINK_COMP_ARRAY(COMP_CODEC("aw882xx_smartpa.0-0034", "aw882xx-aif-0-34")),
    DAILINK_COMP_ARRAY(COMP_PLATFORM("snd-soc-dummy")));
/*如果你需要 IV 反馈，进行如下添加*/
SND_SOC_DAILINK_DEFS(quin_mi2s_tx_smartpa,
    DAILINK_COMP_ARRAY(COMP_CPU("snd-soc-dummy-dai")),
    DAILINK_COMP_ARRAY(COMP_CODEC("aw882xx_smartpa.0-0034", "aw882xx-aif-0-34")),
    DAILINK_COMP_ARRAY(COMP_PLATFORM("snd-soc-dummy")));

```

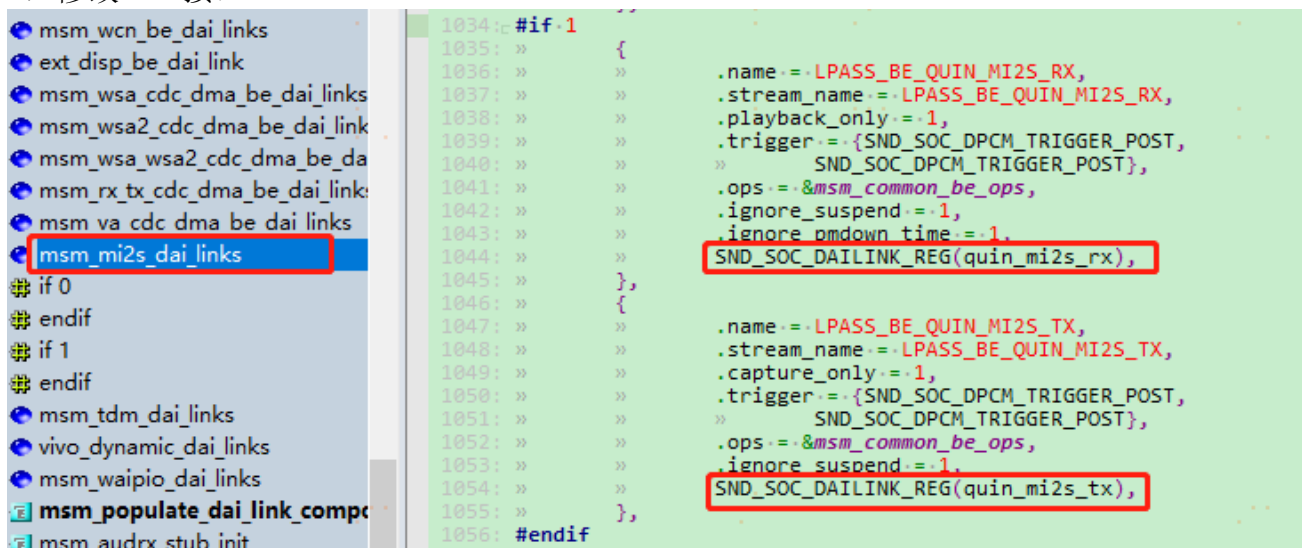
#### 多 PA 配置方法

```

#ifdef CONFIG_SND_SOC_AW882XX
SND_SOC_DAILINK_DEFS(quin_mi2s_rx_smartpa,
    DAILINK_COMP_ARRAY(COMP_CPU("snd-soc-dummy-dai")),
    DAILINK_COMP_ARRAY(COMP_CODEC("aw882xx_smartpa.0-0034", "aw882xx-aif-0-34"),
        COMP_CODEC("aw882xx_smartpa.0-0035", "aw882xx-aif-0-35")),
    DAILINK_COMP_ARRAY(COMP_PLATFORM("snd-soc-dummy")));
/*如果你需要 IV 反馈，进行如下添加*/
SND_SOC_DAILINK_DEFS(quin_mi2s_tx_smartpa,
    DAILINK_COMP_ARRAY(COMP_CPU("snd-soc-dummy-dai")),
    DAILINK_COMP_ARRAY(COMP_CODEC("aw882xx_smartpa.0-0034", "aw882xx-aif-0-34"),
        COMP_CODEC("aw882xx_smartpa.0-0035", "aw882xx-aif-0-35")),
    DAILINK_COMP_ARRAY(COMP_PLATFORM("snd-soc-dummy")));

```

## 2) 修改 DAI 接口

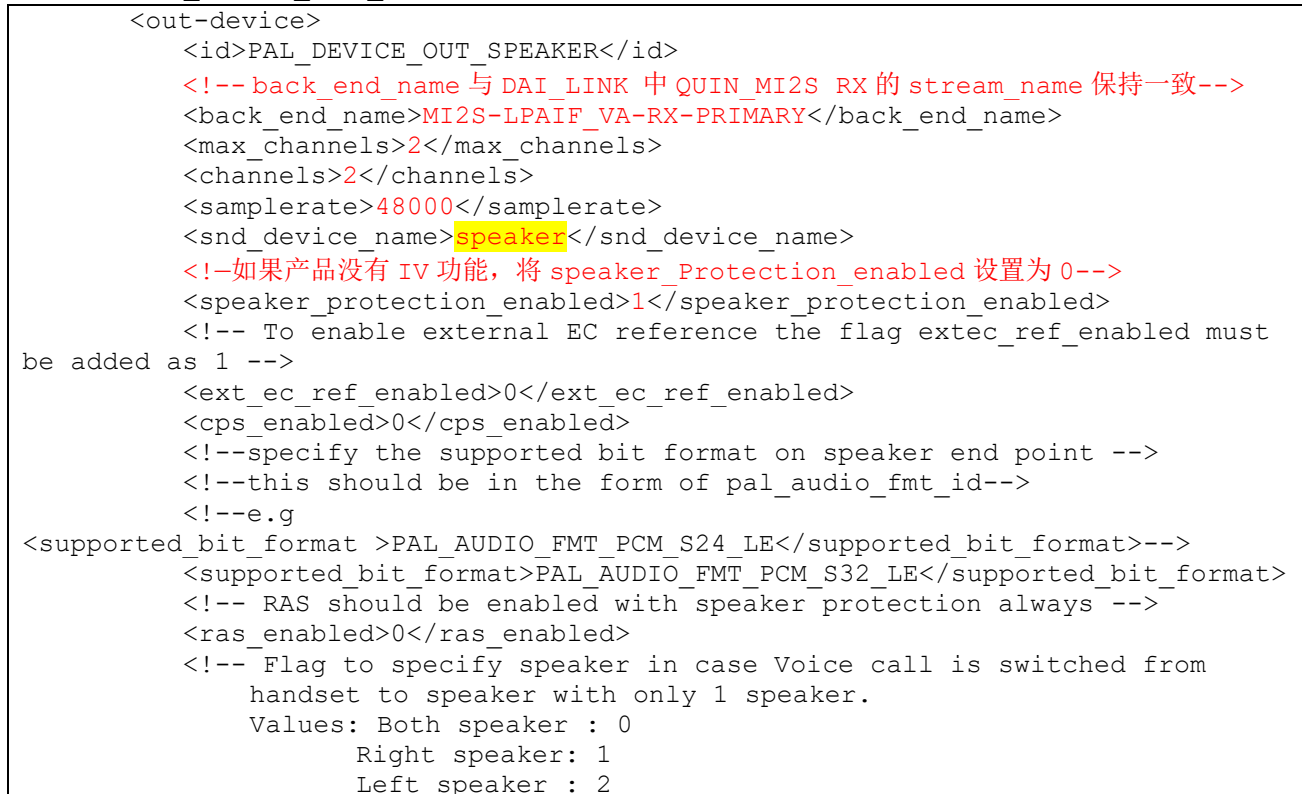


## 2.3 平台通路配置

### 2.3.1 XML 配置

#### Resourcemanager.xml 文件修改

##### 1) 修改 PAL\_DEVICE\_OUT\_SPEAKER 设备的配置



(PS: 如下 stream\_name 对应的字符串为 MI2S-LPAIF\_VA-RX-PRIMARY)

```
{
    .name = LPASS_BE_QUIN_MI2S_RX,
    .stream_name = LPASS_BE_QUIN_MI2S_RX,
    .playback_only = 1,
    .trigger = {SND_SOC_DPCM_TRIGGER_POST,
        SND_SOC_DPCM_TRIGGER_POST},
    .ops = &msm_common_be_ops,
    .ignore_suspend = 1,
    .ignore_pmdown_time = 1,
    SND_SOC_DAILINK_REG(quin_mi2s_rx),
},
{
    .name = LPASS_BE_QUIN_MI2S_TX,
    .stream_name = LPASS_BE_QUIN_MI2S_TX,
    .capture_only = 1,
    .trigger = {SND_SOC_DPCM_TRIGGER_POST,
        SND_SOC_DPCM_TRIGGER_POST},
    .ops = &msm_common_be_ops,
    .ignore_suspend = 1,
    SND_SOC_DAILINK_REG(quin_mi2s_tx),
},
},

#define LPASS_BE_PRI_MI2S_RX "MI2S-LPAIF-RX-PRIMARY"
#define LPASS_BE_PRI_MI2S_TX "MI2S-LPAIF-TX-PRIMARY"
#define LPASS_BE_SEC_MI2S_RX "MI2S-LPAIF_AUD-RX-PRIMARY"
#define LPASS_BE_SEC_MI2S_TX "MI2S-LPAIF_AUD-TX-PRIMARY"
#define LPASS_BE_TERT_MI2S_RX "MI2S-LPAIF-RX-TERTIARY"
#define LPASS_BE_TERT_MI2S_TX "MI2S-LPAIF-TX-TERTIARY"
#define LPASS_BE_QUAT_MI2S_RX "MI2S-LPAIF_RXTX-RX-PRIMARY"
#define LPASS_BE_QUAT_MI2S_TX "MI2S-LPAIF_RXTX-TX-PRIMARY"
#define LPASS_BE_QUIN_MI2S_RX "MI2S-LPAIF_VA-RX-PRIMARY"
#define LPASS_BE_QUIN_MI2S_TX "MI2S-LPAIF_VA-TX-PRIMARY"
#define LPASS_BE_SEN_MI2S_RX "MI2S-LPAIF_WSA-RX-PRIMARY"
#define LPASS_BE_SEN_MI2S_TX "MI2S-LPAIF_WSA-TX-PRIMARY"

#define LPASS_BE_SLIMBUS_0_RX "SLIM-DEV1-RX-0"
#define LPASS_BE_SLIMBUS_0_TX "SLIM-DEV1-TX-0"
#define LPASS_BE_SLIMBUS_1_RX "SLIM-DEV1-RX-1"
#define LPASS_BE_SLIMBUS_1_TX "SLIM-DEV1-TX-1"
#define LPASS_BE_SLIMBUS_2_RX "SLIM-DEV1-RX-2"
#define LPASS_BE_SLIMBUS_2_TX "SLIM-DEV1-TX-2"
```

## Mixer\_paths\_xxx.xml 配置

注释 path name="speaker" 的内容（这里"speaker"与 resourcemanager.xml 中的 snd\_device\_name 对应），I2S 输出无需打开 codec 输出

```
<path name="speaker">
    <!-- name="SmartPA Switch" value="1" /-->
</path>
```

## 2.4 驱动移植有效性验证

按如上操作完成驱动移植，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. TX 驱动移植

#### 3.1 HAL 层移植

按照如下路径中的 patch 文件进行 TX 通路的修改;

```
AW882XX_Driver_QCOM_v1.11.0
├── ap
│   ├── Readme
│   ├── AudioReach
│   │   ├── hardware
│   │   │   └── 0001-Project-AW882XX_AudioReach_TX.patch /*TX移植patch*/
│   │   └── porting_guide
│   │       ├── AW882XX_Android_Driver_Qcom_AudioReach_V1.0.docx
│   │       ├── AW882XX_Android_Driver_Qcom_AudioReach_V1.0.pdf
│   │       ├── AW882XX_Android_Driver_Qcom_AudioReach_V1.0_EN.docx
│   │       ├── AW882XX_Android_Driver_Qcom_AudioReach_V1.0_EN.pdf
│   │       └── Revision.docx
```

#### 3.2 XML 配置

##### Resourcemanager.xml 文件修改

1) 修改 VI 设备的配置 (没有 IV 的产品不需要配置, 产品 IV 支持情况参考 2 章节)

```
<in-device>
  <id>PAL_DEVICE_IN_VI_FEEDBACK</id>
  <!-- back_end_name 与 DAI_LINK 中 QUIN_MI2S_TX 的 stream_name 保持一致-->
  <back_end_name>MI2S-LPAIF_VA-TX-PRIMARY</back_end_name>
  <max_channels>2</max_channels>
  <channels>2</channels>
  <snd_device_name>vi-feedback</snd_device_name>
  <supported_bit_format>PAL_AUDIO_FMT_PCM_S32_LE</supported_bit_format>
</in-device>
```

(PS: 如下 stream\_name 对应的字符串为 MI2S-LPAIF\_VA-TX-PRIMARY)

```
#if 1
{
.name = LPASS_BE_QUIN_MI2S_RX,
.stream_name = LPASS_BE_QUIN_MI2S_RX,
.playback_only = 1,
.trigger = {SND_SOC_DPCM_TRIGGER_POST,
SND_SOC_DPCM_TRIGGER_POST},
.ops = &msm_common_be_ops,
.ignore_suspend = 1,
.ignore_pmdown_time = 1,
SND_SOC_DAILINK_REG(quin_mi2s_rx),
},
{
.name = LPASS_BE_QUIN_MI2S_TX,
.stream_name = LPASS_BE_QUIN_MI2S_TX,
.capture_only = 1,
.trigger = {SND_SOC_DPCM_TRIGGER_POST,
SND_SOC_DPCM_TRIGGER_POST},
.ops = &msm_common_be_ops,
.ignore_suspend = 1,
SND_SOC_DAILINK_REG(quin_mi2s_tx),
},
},
#endif

#define LPASS_BE_PRI_MI2S_RX "MI2S-LPAIF-RX-PRIMARY"
#define LPASS_BE_PRI_MI2S_TX "MI2S-LPAIF-TX-PRIMARY"
#define LPASS_BE_SEC_MI2S_RX "MI2S-LPAIF-AUD-RX-PRIMARY"
#define LPASS_BE_SEC_MI2S_TX "MI2S-LPAIF-AUD-TX-PRIMARY"
#define LPASS_BE_TERT_MI2S_RX "MI2S-LPAIF-RX-TERTIARY"
#define LPASS_BE_TERT_MI2S_TX "MI2S-LPAIF-TX-TERTIARY"
#define LPASS_BE_QUAT_MI2S_RX "MI2S-LPAIF-RXTX-RX-PRIMARY"
#define LPASS_BE_QUAT_MI2S_TX "MI2S-LPAIF-RXTX-TX-PRIMARY"
#define LPASS_BE_QUIN_MI2S_RX "MI2S-LPAIF-VA-RX-PRIMARY"
#define LPASS_BE_QUIN_MI2S_TX "MI2S-LPAIF-VA-TX-PRIMARY"
#define LPASS_BE_SEN_MI2S_RX "MI2S-LPAIF-WSA-RX-PRIMARY"
#define LPASS_BE_SEN_MI2S_TX "MI2S-LPAIF-WSA-TX-PRIMARY"

#define LPASS_BE_SLIMBUS_0_RX "SLIM-DEV1-RX-0"
#define LPASS_BE_SLIMBUS_0_TX "SLIM-DEV1-TX-0"
#define LPASS_BE_SLIMBUS_1_RX "SLIM-DEV1-RX-1"
#define LPASS_BE_SLIMBUS_1_TX "SLIM-DEV1-TX-1"
#define LPASS_BE_SLIMBUS_2_RX "SLIM-DEV1-RX-2"
#define LPASS_BE_SLIMBUS_2_TX "SLIM-DEV1-TX-2"
#define LPASS_BE_SLIMBUS_3_RX "SLIM-DEV1-RX-3"
#define LPASS_BE_SLIMBUS_3_TX "SLIM-DEV1-TX-3"
```

## 4. 校准和低压保护功能

校准和低压保护功能请参考移植包中《smartpa\_cali\_and\_monitor》目录下移植文档。

## 5. 调试接口

### 5.1 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

5.2 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

Awinic Driver