AW883XX Android Driver(MTK)

版本： V2.1

时间： MAY 25, 2022

contents

1. INFORMATION

Driver File	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
Smart PA	aw88394、aw88395
I²C Address	0x34/0x35/0x36/0x37
Platform	mt6853

2. DRIVER PORTING

2.1 SmartPA Configuration

2.1.1 Add PA Configuration

Add aw883xx smartpa in ProjectXXX.mk

```
MTK_AUDIO_SPEAKER_PATH = smartpa_awinic_aw883xx
```

Configuring the above options will be displayed The following parameters when Overall compilation in AudioParamOptions.xml

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

Add Configuration in SmartPa_AudioParam.xml

```
--- a/audio_param/SmartPa_AudioParam.xml
+++ b/audio_param/SmartPa_AudioParam.xml
@@ -8,6 +8,7 @@
     <Param path="smartpa_cirrus_cs35135" 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>
```

Add description of aw883xx in SmartPa_ParamUnitDesc.xml：

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

2.2 AW883XX Driver Porting

2.2.1 DTS Configuration

Mono PA Configuration

```
&i2c_x {                                          /*x : I2C bus*/
+       aw883xx_smartpa_0: aw883xx_smartpa@34 {  /* 0x34 : I2C Address*/
+           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>;            /*Minimum calibration value (mohms)*/
+           re-max= <40000>;            /*Maximum calibration value (mohms)*/
+           status = "okay";
+       };
       /*re means resistance of speaker */
```

Multi PA configuration

```
&i2c_x {
+       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>;      /*The first PA configured as 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>;      /* The Second PA configured as 1*/
+            re-min = <1000>;
+            re-max= <40000>;
+           status = "okay";
+      };
+};
/*The above is an example of double Pa. when there are multiple PAS, the sound channel serial number increases in turn. For other attributes, refer to the description of the Mono PA configuration */
```

2.2.2 Driver Configuration

Kernel Compile Configuration

defconfig Configuration：

```
CONFIG_SND_SMARTPA_AW883XX=y
```

Create aw883xx directory in the kernel codecs directory and add driver files:

```
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 Configuration:

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

Makefile Configuration:

```
#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 Compile Configuration

If select the KO compilation modes, add configuration are as follows:


```
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 Configuration

PA needs to configure register parameters to work normally. The configuration steps of PA bin file are as follows:

Compile Configuration

Add the bin file compilation option at the corresponding location of the platform：

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

/*xxxx means Platform path*/
```

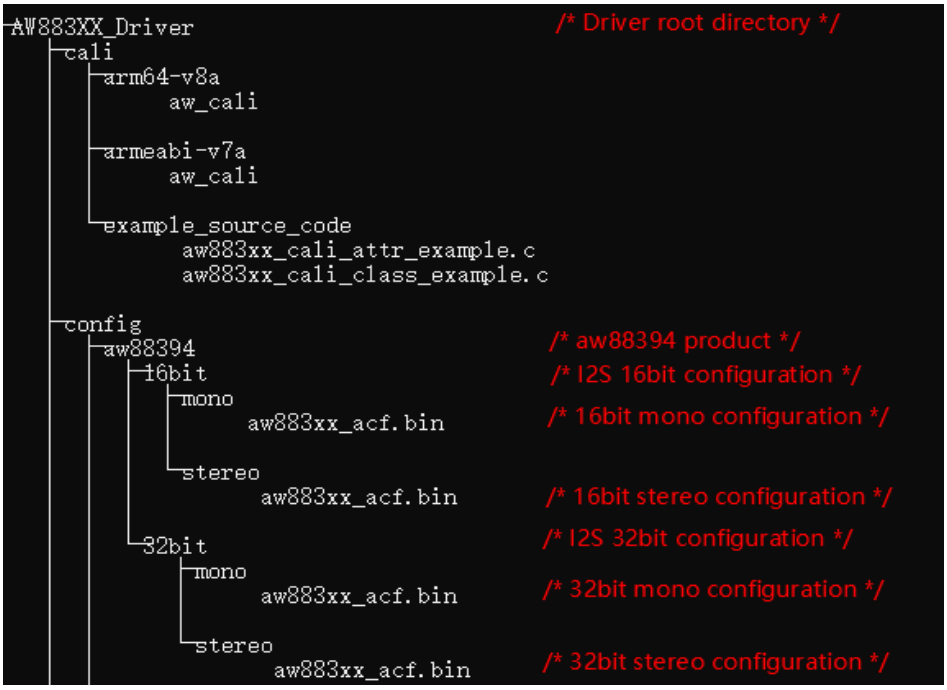
Path Configuration

Add the directory of bin file in the firmware_class.c, the general directory is vendor / firmware

```
static const char * const fw_path[] = {
    fw_path_para,
    "/vendor/firmware",
    "/lib/firmware/updates/" UTS_RELEASE,
    "/lib/firmware/updates",
    "/lib/firmware/" UTS_RELEASE,
    "/lib/firmware"
};
/* Note: In the debugging stage, you can directly push the bin file to / vendor / firmware / */
```

Bin selection

Select the bin according to the platform I2S output format and PA quantity, taking aw88394 as an example：



2.3 Platform Driver Configuration

2.3.1 DAI_LINK Configuration

Different platform dai_link configuration is as follows:

4G Platform Configuration

1）Add awinic_codecs array:

Mono PA Configuration

```
struct snd_soc_dai_link_component awinic_codecs[] = {
    {
        .of_node = NULL,
        .dai_name = "aw883xx-aif-6-34",         /*I2C BUS:6, I2C Address:34*/
        .name = "aw883xx_smartpa.6-0034",
    },
};
```

Multi PA configuration

```
struct snd_soc_dai_link_component awinic_codecs[] = {
    {
        .of_node = NULL,
        .dai_name = "aw883xx-aif-6-34",         /*I2C BUS:6, I2C Address:34*/
        .name = "aw883xx_smartpa.6-0034",
    },
    {
        .of_node = NULL,
        .dai_name = "aw883xx-aif-6-35",         /*I2C BUS:6, I2C Address:35*/
```



```
        .name = "aw883xx_smartpa.6-0035",
    },
};
/*The above example is the dual PA configuration. When there are multiple PAS,
the configuration is added in turn*/
```

2） Modify DAI interface

```
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 Platform Configuration

Mono PA Configuration

```
        mtk_spk_i2s_in = <0>;
        /* mtk_spk_i2s_mck = <3>; */
        mediatek,speaker-codec {
-           sound-dai = <&speaker_amp>;
+           sound-dai = <&aw883xx_smartpa_0>;
        };
    };
/* Add Dai information according to DTS node < aw883xx_smartpa_0 > */
```

Multi PA configuration

```
        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>;
        };
    };
 /* Take two PA as an example, where the information corresponding to I2C node
is < aw883xx_smartpa_0>、<aw883xx_smartpa_1> */
```

2.4 I2S Routes Configuration

Only For 5G Platform

```
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 Drive Verification

The above operations have completed the driver integration. Confirm that is effective through the following driver log.

I2C communication succeeded:

```
[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
```

Sound card registration succeeded：

```
[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]
```

PA Bin loaded successfully：

```
[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:0xfffffdf2
[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
```

Play music and PA makes sound：

```
[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. DRIVE CALIBRATION FUNCTION

3.1 Purpose Of Calibration

For speaker protection requirements, the aw883xx driver supports speaker calibration on the production line. After calibration, the re value of the qualified speaker is written into the persistent partition of the mobile phone. When the chip configuration is loaded at startup, the driver will write the calibration value read in the persist partition into the chip DSP to achieve the function of temperature protection.

3.2 Calibration Adaptation

3.2.1 Calibration Result Saving Path Adaptation

1）The aw883xx drive defines the path in aw883xx_calib.c to save the calibration value file as follows:

```
/*******************************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）Please confirm whether the same path exists in the mobile phone. If there is no corresponding path, the calibration will fail：

```
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 Calibration Mode

Aw883xx driver provides misc, class and attr calibration methods.

3.3.1 Misc Calibration

The Misc calibration of aw883xx driver is realized through executable file. Follow the steps below：

Get executable

```
aw883xx_driver                  /*driver root directory*/
   ├cali
   │  ├arm64-v8a
   │  │     aw_cali             /* 64bit system executable*/
   │  │
   │  ├armeabi-v7a
   │  │     aw_cali             /* 32bit system executable*/
   │  │
   │  └example_source_code
   │        aw883xx_cali_attr_example.c
   │        aw883xx_cali_class_example.c
   │        readMe.txt
```

Configuration executable

```
C:\Users\ >adb push aw_cali system/bin/
aw_cali: 1 file pushed. 0.3 MB/s (14384 bytes in 0.049s)

C:\Users\ >adb shell chmod 0777 system/bin/aw_cali
```

Command introduction

```
msm8996:/ # aw_cali
 Calibration executables version: v0.1.1
------------------------------------------------------------
./aw_cali [dev_name] start_cali
./aw_cali dev0 start_cali
------------------------------------------------------------
./aw_cali [dev_name] cali_re
./aw_cali dev0 cali_re
------------------------------------------------------------
./aw_cali [dev_name] cali_f0
./aw_cali dev0 cali_f0
------------------------------------------------------------
./aw_cali [dev_name] get_spkr_status
./aw_cali dev0 get_spkr_status
------------------------------------------------------------
./aw_cali [dev_name] set_cali_re re_value1 [re_value2]
./aw_cali dev0 set_cali_re 8000
------------------------------------------------------------
./aw_cali [dev_name] get_re_range
./aw_cali dev0 get_re_range
------------------------------------------------------------
```

calibration cmd

re calibration

f0 calibration

get real time status

set calibration value of re

calibration re range

Parameter explanation (Note: [ ] means this option can be left blank)

dev_name	Calibrate single device “devx”, x corresponds to the sound-channel configured in dts, When node is not configurated, the application can be automatically searched.

Calibration Steps

1) Play mute music;
2) Start the calibration:


```
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 Calibration

Node Function

nodes	function
/sys/calss/smartpa/cali_time	1. Set calibration time 2.Show current calibration time
/sys/calss/smartpa/f0_calib	Calibrate f0
/sys/calss/smartpa/re25_calib	1. Calibrate re 2. Save calibration re value
/sys/calss/smartpa/f0_q_calib	Calibrate f0 and q
/sys/calss/smartpa/re_range	Show re_range

Calibration Steps

1) Play mute music;
2) Re calibration:

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

f0 calibration:

```
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 Calibration

Node Path

6-0034:“6”means I2C bus，“0034”meand I2C address：

```
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 #
```


Node Function

nodes	function
cali_time	1. Set calibration time 2.Show current calibration time
cali_re	1. Calibrate re 2. Save calibration re value
cali_f0	Calibrate f0
cali_f0_q	Calibrate f0 and q
re_range	Show re_range

Calibration Steps

1）play mute file；
2）Re calibration：

```
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 #
```

f0 calibration :

```
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 Verification Of Calibration Results

1）Calibrate several times to confirm whether the calibration re is within the effective range and the value will change slightly. (re effective range confirmed with hardware colleagues)，Calibration twice in attr mode, for example:

```
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）Check whether the re value is written to the file：

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

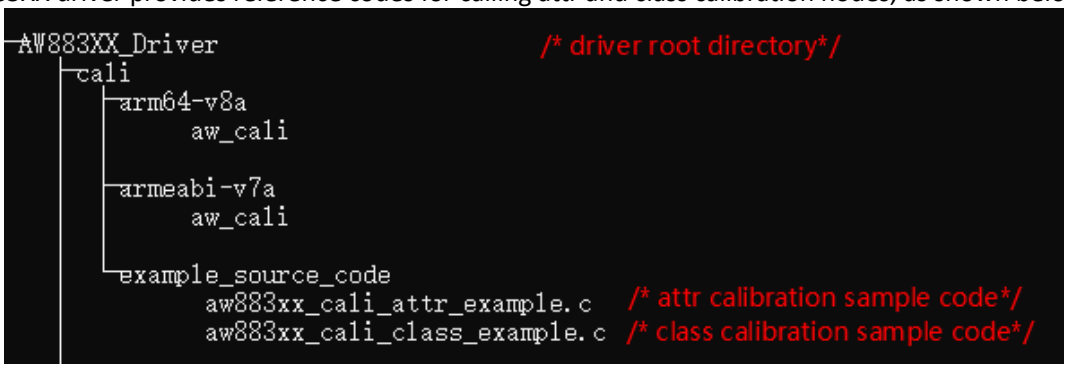
3）When playing music, check the DSP_ Re node, confirm that it is the same as the above calibration value：

（Note: the above difference is the normal error of fixed-point calculation results, and the general range is 1）

4）Restart the phone, play music and check the DSP again_ Re node, confirm DSP_ The value in the re node is the same as the re value in the file。

3.5 Calibration Example Code

AW883XX driver provides reference codes for calling attr and class calibration nodes, as shown below.



4.DEBUG INTERFACE

4.1 Device node

AW883XX Driver will create multiple device node files with different functions. The path is sys/bus/i2c/drivers/aw883xx_smartpa/*-00xx, where * is the i2c bus number and xx is the i2c address. You can use adb to read and write nodes to debug AW883XX Driver.

reg

Node name	reg
Description	read and write all register value of aw883xx
Instruction	read register value：cat reg write register value：echo reg_addr reg_data > reg (Hexadecimal operation)
Example	cat reg （get all the values of the register with read permission）

	echo 0x04 0x0241 > reg （write the value of 0x0241 to the register of 0x04）

rw

Node name	rw
Description	read and write single register value of aw883xx
Instruction	read register value：echo reg_addr > rw (Hexadecimal operation) cat rw write register value：echo reg_addr reg_data > rw (Hexadecimal operation)
Example	echo 0x04 > rw (read the value of the 0x04 register) cat rw echo 0x04 0x0241 > rw (write the value of 0x0241 to the register of 0x04)

drv_ver

Node name	drv_ver
Description	get driver version
Instruction	get driver version：cat drv_ver

dsp_rw

Node name	dsp_rw
Description	read and write single dsp register value of aw883xx
Instruction	read dsp register： echo reg_addr > dsp_rw （Hexadecimal operation） cat dsp_rw write dsp register： echo reg_addr reg_data > dsp_rw （Hexadecimal operation）
Example	echo 0x8601 > dsp_rw （Read dsp register 0x8604） cat dsp_rw echo 0x8604 0x4011 > dsp_rw （Write 0x4011 to dsp register 0x8604）

dsp

Node name	dsp
Description	Get dsp firmware and dsp config info
Instruction	Get dsp firmware and dsp config： cat dsp

Example	cat dsp

fade_step

Node name	fade_step
Description	set step of fade in and fade out
Instruction	set step echo step > fade_step get step cat fade_step
Example	echo 6 > fade_step （set the step to 6） cat fade_step (get the current step of fade in and fade out)

dbg_prof

Node name	dbg_prof
Description	scene switching function control node
Instruction	switch scene function enable echo 1 > dbg_prof switch scene function disable echo 0 > dbg_prof

fade_en

Node name	fade_en
Description	fade in and fade out function control node
Instruction	fade in and fade out enable echo 1 > fade_en fade in and fade out disable echo 0 > fade_en

monitor

Node name	monitor
Description	Software Monitor function control node
Instruction	monitor enable echo 1 > monitor monitor disable echo 0 > monitor

monitor_update

Node name	monitor_update
Description	monitor config update control node
Instruction	update monitor config echo 1 > monitor_update

dsp_re

Node name	dsp_re
Description	get re value of dsp
Instruction	cat dsp_re

i2c_log_en

Node name	I2c_log_en
Description	I2C data printing control node
Instruction	i2c data printing enable echo 1 > i2c_log_en i2c data printing disable echo 0 > i2c_log_en get node cat print_dbg

phase_sync

Node name	phase_sync
Description	phase synchronization function control node
Instruction	phase synchronization function enable echo 1 > phase_sync phase synchronization function disable echo 0 > phase_sync get phase_sync status cat phase_sync

spk_temp

Node name	spk_temp
Description	display the real-time status of the speaker
Instruction	cat spk_temp

cali_re

Node name	cali_re
Description	Calibration re; Set re value
Instruction	Calibration re: cat cali_re set re value: echo dev[0]:6848 dev[1]:6683 > cali_re (Take the stereo PA configuration as an example. When there are multiple PAS, it is added according to the format)

cali_f0

Node name	cali_f0

Description	Calibration f0
Instruction	Calibration f0: cat cali_f0

cali_f0_q

Node name	cali_f0_q
Description	Calibration f0,q
Instruction	Calibration f0,q: cat cali_f0_q

cali_time

Node name	cali_time
Description	Read Calibration delay time Set Calibration delay time
Instruction	Read Calibration delay time： cat cali_time Set Calibration delay time： echo 3000 > cali_time (Unit: ms)

re_range

Node name	re_range
Description	Read re Calibration range
Instruction	Read re Calibration range： cat re_range

4.2 Kcontrol

"x" represents the device number

aw_dev_x_switch

Name	aw_dev_x_switch
Description	PA switch
Instruction	tinymix aw_dev_ x _switch Enable the device x of PA allow to turn on tinymix aw_dev_ x _switch Disable The device x of PA is not allowed to be turned on

aw_dev_x_prof

Name	aw_dev_x_prof
Description	scene switch (assuming that Music and Receive scenes are configured in the bin file)
Instruction	tinymix aw_dev_ x _prof Music the device x of PA switch scene to Music tinymix aw_dev_ x _prof Receive the device x of PA switch scene to Receive

aw_dev_x_monitor_switch

Name	aw_dev_x_monitor_switch
Description	PA monitor switch
Instruction	tinymix aw_dev_ x _switch Enable the device x of PA allow to turn on monitor tinymix aw_dev_ x _switch Disable the device x of PA is not allowed to be turned on

aw883xx_fadein_us

Name	aw883xx_fadein_us
Description	fade in step time interval setting
Instruction	tinymix aw883xx_fadein_us 500 set 500us fade in step time interval

aw883xx_fadeout_us

Name	aw883xx_fadeout_us
Description	fade out step time interval setting
Instruction	tinymix aw883xx_fadeout_us 500 set 500us fade out step time interval

5 APPENDIX

5.1 PLATFORM I2C BUS DYNAMIC CHANGES

If the platform will change the i2c bus number, the i2c bus number of dynamic changes will cause the failure of dai_link matching, can solve by modifying the configuration in the device tree and dai_link configuration.

5.1.1 DTS Configuration

Add the rename-flag attribute to the driver node and set the attribute value to 1.

Mono PA Configuration

```
&i2c_x {                                                    /*x : I2C bus*/
+       aw883xx_smartpa_0: aw883xx_smartpa@34 {  /* 0x34 : I2C Address*/
+           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>;              /*Minimum calibration value (mohms)*/
+           re-max= <40000>;              /*Maximum calibration value (mohms)*/
+           rename-flag= <1>;
+           status = "okay";
+       };
       /*re means resistance of speaker */
```

Multi PA configuration

```
&i2c_x {
+       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>;       /*The first PA configured as 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>;      /* The Second PA configured as 1*/
+            re-min = <1000>;
+            re-max= <40000>;
+            rename-flag= <1>;
```



```
+           status = "okay";
+       };
+};
/*The above is an example of double Pa. when there are multiple PAS, the sound channel serial number increases in turn. For other attributes, refer to the description of the Mono PA configuration */
```

5.1.2 DAI_LINK Configuration

Change the suffix of codec_name and codec_dai_name to sound-channel,different platform dai_link configuration is as follows:

4G Platform Configuration

1) Add awinic_codecs array:

Mono PA Configuration

```
struct snd_soc_dai_link_component awinic_codecs[] = {
    {
        .of_node = NULL,
        .dai_name = "aw883xx-aif-0",          /* Change the suffix to sound-channel of DTS node */
        .name = "aw883xx_smartpa_0",
    },
};
```

Multi PA configuration

```
struct snd_soc_dai_link_component awinic_codecs[] = {
    {
        .of_node = NULL,
        .dai_name = "aw883xx-aif-0",        /* Change the suffix to sound-channel of DTS node */
        .name = "aw883xx_smartpa_0",
    },
    {
        .of_node = NULL,
        .dai_name = "aw883xx-aif-1",        /* Change the suffix to sound-channel of DTS node */
        .name = "aw883xx_smartpa_1",
    },
};
/*The above example is the dual PA configuration. When there are multiple PAS, the configuration is added in turn*/
```

2) Modify DAI interface

```
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 Platform Configuration

5G platforms need no additional modifications.

