

AW882XX AudioReach Calibration and low pressure protection Porting guidelines

Version: V1.2

Time: March 2025

1. Brief

This document mainly introduces the AW882XX AudioReach calibration code, calibration executable file and low temperature and low voltage protection related code, designed to help customers use aw882xx chip in AudioReach platform.

1.1 Edit record

Versions	Edit note
V1.0	First release
V1.1	1. Add a section on pre-migration requirements 2. Optimize the calibration directory structure
V1.2	1. Add instructions for setting offset

2. Low temperature and low pressure protection function

2.1 Purpose

This feature is used in the Android Reach architecture to avoid problems such as noise at low temperatures and low pressures.

2.2 Precondition

1. The following attribute values must be configured in the AW882xx driver:

```
monitor-mode = "hal_monitor";
```

2. AWINIC algorithm can work normally and the device makes sound normally.

2.3 Porting steps

- a. Create the awinic_ar directory in the project source /pal/device/ directory
- b. Copy the files in the code folder to the pal/device/awinic_ar/ directory
- c. Refer to the attachment <Android_mk_for_monitor.patch> to add compilation options to Android.mk in the pal directory
- d. Refer to the attachment <monitor_pal.patch> to modify the code of the Pal part
- e. Modify agm codes by referring to <params_agm.patch>

2.4 Interface description

Function name	Function
aw_audioreach_monitor_init(struct aw_dev_info *dev_info)	initialize
aw_audioreach_monitor_deinit()	Release resources
aw_audioreach_monitor_start()	Enabling the monitor function
aw_audioreach_monitor_stop()	Disable the monitor function

2.5 Parameter description

struct aw_dev_info structure parameters are described as follows

Parameter	Type	Description
virt_mixer	struct mixer *	Virtual mixer
hw_mixer	struct mixer *	Physics mixer

2.6 Validity verification

The kernel log is obtained during playback. If the following keywords are displayed, the function is normal

```

10:38:18.251 1814 1814 I [Awinic] [0-0035]aw_monitor_get_temperature: reg val is 0x001f
10:38:18.251 1814 1814 I [Awinic] [0-0035]aw_monitor_get_temperature: chip temperature = 31
10:38:18.259 1814 1814 I [Awinic] [0-0035]aw882xx_hal_monitor_work_get: vmax is 0x0
10:38:19.273 1814 1814 I [Awinic] [0-0035]aw882xx_monitor_get: monitor switch is 1
10:38:19.279 1814 1814 I [Awinic] [0-0035]aw_monitor_get_voltage: chip voltage is 4075
10:38:19.280 1814 1814 I [Awinic] [0-0035]aw_monitor_get_temperature: reg val is 0x001f
10:38:19.280 1814 1814 I [Awinic] [0-0035]aw_monitor_get_temperature: chip temperature = 31
10:38:19.280 1814 1814 I [Awinic] [0-0035]aw882xx_hal_monitor_work_get: vmax is 0x0
10:38:20.284 1814 1814 I [Awinic] [0-0035]aw882xx_monitor_get: monitor switch is 1
10:38:20.286 1814 1814 I [Awinic] [0-0035]aw_monitor_get_voltage: chip voltage is 4087
10:38:20.287 1814 1814 I [Awinic] [0-0035]aw_monitor_get_temperature: reg val is 0x001f
10:38:20.287 1814 1814 I [Awinic] [0-0035]aw_monitor_get_temperature: chip temperature = 31
10:38:20.287 1814 1814 I [Awinic] [0-0035]aw882xx_hal_monitor_work_get: vmax is 0x0

```

3. Calibrate the function Porting guide

Awinic provides two calibration schemes in the form of functional calibration interfaces and executable files. The appropriate calibration scheme can be selected according to the project needs.

3.1 The way a function interfaces

3.1.1 Function interface description

The interface function is declared in the aw_ar_api.h file and has the following meanings:

```
/*calibrate re*/
int aw_audioreach_dsp_read_r0(struct aw_dev_info *dev_info, int *r0, int dev_num);

/*calibrate f0*/
int aw_audioreach_dsp_read_f0(struct aw_dev_info *dev_info, int *f0, int dev_num);

/*calibrate f0 and q*/
int aw_audioreach_dsp_read_f0_q(struct aw_dev_info *dev_info, int *f0, int *q, int dev_num);

/*set ret*/
int aw_audioreach_dsp_set_re(struct aw_dev_info *dev_info, int *cali_re, int dev_num);

/*Get real-time re and horn temperature*/
int aw_audioreach_dsp_read_st(struct aw_dev_info *dev_info,
                             int *r0, int *te, int dev_num);

/*Gets the range of re*/
int aw_audioreach_dsp_get_re_range(int *re_min, int *re_max, int dev_num);

/*set algo en*/
int aw_audioreach_dsp_set_algo_en(struct aw_dev_info *dev_info, unsigned int is_enable);

/*Set re to disk file*/
int aw_audioreach_set_re_to_file(int *cali_re, int dev_num);

/*ger re from disk file*/
int aw_audioreach_get_re_from_file(int *cali_re, int dev_num);
```

3.1.2 Parameter description

Parameter description	Type	Description
struct aw_dev_info *dev_info	struct aw_dev_info { struct mixer *virt_mixer; struct mixer *hw_mixer; };	1.virt_mixer: mixer of virtual sound card; 2.hw_mixer: mixer of physical sound card;
int32_t *r0	int32_t *	To get the head pointer of the buff for calibrating re, make space based on the actual PA number for calibrating re values
int32_t *f0	int32_t *	To get the head pointer of the buff that calibrated f0, make space based on the number of PA that actually calibrated the f0 value
int32_t *te	int32_t *	To get the head pointer of the buff that calibrates te, open up space according to the actual PA number that gets te values
int32_t *cali_re	int32_t *	The head pointer to the buff that holds the cali_re value
int32_t dev_num	int32_t	The number of PA or the channel of PA
int *re_min	int *	To obtain the buff header pointer with the minimum effective range of re, open up space based on the actual amount of PA obtained

Int *re_max	int *	To get the buff head pointer with the maximum effective range of re, open up space based on the actual amount of PA obtained
-------------	-------	--

3.1.3 Calibration steps

1. Play mute file
2. Call the function interface for calibrating re and calibrating f0
3. Get the calibration values re and f0, judge the reasonableness of the values, if reasonable, save the re to a disk file
4. Stop playback and calibration is over

3.1.4 Set re to AWINIC algo

After Re calibration is completed, the calibrated re value needs to be set to the awinic algorithm for each playback.

Please refer to the specific transplantation 《set_re_pal.patch》

The default is to use the following interface provided by awinic to get re from a file. To customize, modify or replace the following functions.

```
ret = aw_audioreach_get_re_from_file(cali_re, numberOfChannels);
```

Note: If using chips AW88460/AW88461/AW88698, an additional offset setting is required. Please refer to the specific transplantation 《set_offset_pal.patch》

3.1.5 Verification of calibration validity

- 1) Calibrate multiple times to confirm whether the calibration re is within the effective range and the value is not constant
- 2) Grab the dsp log, start playing music, and verify that the re value set is the same as the above calibration value:

```
[Awinic] [D] aw_sToneEvalSetCalRe: Algo.cali_re := set.cali_re + ra := .8560 + .449 := .9010 mOhm .  
9315d3a205b4177696e69635d5b445d2061775f73546f6e654576616c53657443616c52653a20416c676f2063616c69207265203d
```

- 3) Restart the phone and repeat step 2 to check whether it is normal

3.2 Calibrate the executable file

awinic uses the calibration interface to write the calibration executable file

3.2.1 Migration specification

- a. Refer to Android_mk_for_cali_exec.patch to compile the source code into an executable file
- b. Depending on the project, modify the following compilation options

```
Change the red field according to back_end_name corresponding to  
PAL_DEVICE_OUT_SPEAKER in resourcemanager.xml  
+LOCAL_CFLAGS += -DAW_BACK_END_NAME=\"MI2S-LPAIF VA-RX-PRIMARY\"
```

```
Fill in all PCM-devices that support playback according to card-defs.xml
```

```
+LOCAL CFLAGS += -
DAW_PCM_NAME_LIST="\PCM100,PCM101,PCM102,PCM103,PCM104,COMPRESS105,VOICEMODE1p
,VOICEMODE2p,VOICEMODE1c,VOICEMODE2c,PCM110,PCM111,PCM112,PCM113,PCM114,PCM1
15,PCM116,PCM117\"
+
```

3.2.2 Calibration command

```
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*/
```

Parameter interpretation(Note: [] indicates this option is optional)

dev_name	Used to calibrate individual devices, the dev_name used by different devices corresponds to the sound-channel configured in the dts as follows: 0: aw882xx_smartpa_l 1: aw882xx_smartpa_r 2: aw882xx_smartpa_sec_l 3: aw882xx_smartpa_sec_r If this parameter is not specified, all devices are calibrated by default.
noise	The noise parameter is used to play white noise when calibrating f0. If the customer plays white noise for calibration, you do not need to set this parameter.
cali_re_time	The time used to set the calibration re value, in ms, shall not be less than 1000ms; If this parameter is not specified, the default calibration time is 3000ms.
re_value1	The calibration resistance value (unit: mohm) that needs to be set to the system.
params_file	Path corresponding to the algorithm parameter.

3.2.3 Calibration steps

- a. Play mute file
- b. Start-up calibration

3.2.4 Set re to AWINIC algo

Refer to: **3.1.4**

3.2.5 Verification of calibration validity

Refer to: **3.1.5**

AWinic Driver