# AW87XXX Android Driver(MTK)



版本： V2.8

时间： 2021 年 11 月

 

# 修订记录

| 日期 | 版本 | 描述 | 作者 |
|---|---|---|---|
| 2021-03 | V2.0 | 驱动兼容 | 赵忠波 |
| 2021-06 | V2.1 | 兼容产品 aw87418 | 赵忠波 |
| 2021-06 | V2.2 | 兼容产品 aw87319 | 赵忠波 |
| 2021-06 | V2.3 | 增加 esd 检测 | 赵忠波 |
| 2021-07 | V2.4 | 增加 4G/5G 平台函数调用切场景 | 赵忠波 |
| 2021-07 | V2.5 | 优化部分描述 | 赵忠波 |
| 2021-09 | V2.6 | 兼容产品 aw87390 | 周慧栋 |
| 2021-10 | V2.6.1 | 修改场景字符串文字描述 | 周慧栋 |
| 2021-11 | V2.7 | 修改 acf 支持场景的描述 | 赵忠波 |
| 2021-11 | V2.8 | 兼容产品 aw87418 | 赵忠波 |



 

# 1. 目录





 

# AW87XXX Android Driver

## 1. Information

| | |
|---|---|
| **Driver file** | aw87xxx.c, aw87xxx.h, aw_device.c, aw_device.h, aw_monitor.c, aw_monitor.h, aw_dsp.c, aw_dsp.h, aw_acf_bin.c, aw_acf_bin.h, aw_log.h, aw_bin_parse.c, aw87xxx_pid_39_reg.h, aw87xxx_pid_59_3x9_reg.h, aw87xxx_pid_59_5x9_reg.h, aw87xxx_pid_5a_reg.h, aw87xxx_pid_18_reg.h, aw87xxx_pid_9b_reg.h, aw87xxx_pid_76_reg.h |
| **Support product** | aw87319,<br>aw87329, aw87339, aw87349,<br>aw87359, aw87389,<br>aw87509, aw87519, aw87529, aw87539,<br>aw87549, aw87559,aw87569, aw87579, aw81509<br>aw87390,<br>aw87418 |
| **Support I²C** | aw87418:0x5C<br>other:0x58, 0x59, 0x5A, 0x5B |
| **ADB debug** | Yes |

## 2. Project Config

```
#add aw87xxx smartpa
CONFIG_SND_SOC_AW87XXX=y
```

## 3. Kernel Driver

### 3.1 AW87XXX Smart K PA Driver

#### 3.1.1 dtsi 配置

基于设备树的 I2C 设备驱动需要在 kernel 添加 aw87xxx 的设备树配置。

注意：AW87359/AW87389/AW87549/AW87390 产品没有复位引脚，无需配置 reset-gpio。
dev_index 是对 PA 设置的编号，可设置为 0/1/2/3 等，此参数可不配置，默认按照 PA 的注册顺序进行编号，驱动会以 dev_index 作为默认参数的解析依据。

下表是各产品对应的 CHIPID 和是否带有 reset-pin 信息。

| 产品 | CHIPID | reset-pin |
|---|---|---|
| AW87319 | 0x9B | Y |
| AW87329/AW87339/AW87349 | 0x39 | Y |
| AW87519/AW87529 | 0x59 | Y |
| AW87359/AW87389 | 0x59 | N |
| AW87549 | 0x5A | N |
| AW87559/AW87569/AW87579/AW81509 | 0x5A | Y |
| AW87390 | 0x76 | N |
| AW87418 | 0x18 | Y |

在 kernel/arch/arm/boot/dts/*.dtsi 文件添加 aw87xxx 的配置。
**单 PA 配置：**

 

```
    &i2c_x {     /*x 表示对应的总线号*/
        /* AWINIC AW87XXX Smart K PA */
        aw87xxx_pa@58 {
                    compatible = "awinic,aw87xxx_pa";
                    reg = <0x58>;
                    reset-gpio = <&pio 1 0>; /*带 reset 引脚的配置*/
                    dev_index = < 0 >;
                    status = "okay";
        };
        /* AWINIC AW87XXX Smart K PA End */
    };
```

如需要使用 ESD 检测功能，请在 dtsi 配置中增加配置 esd-enable = “true”;(true 为使能，false 为不使能)

**双 PA 配置：**

```
    &i2c_x {     /*x 表示对应的总线号*/
        /* AWINIC AW87XXX Smart K PA */
        aw87xxx_pa@58 {
                    compatible = "awinic,aw87xxx_pa";
                    reg = <0x58>;
                    reset-gpio = <&pio 1 0>; /*带 reset 引脚的配置*/
                    dev_index = < 0 >;
                    status = "okay";
        };
        aw87xxx_pa@59 {
                    compatible = "awinic,aw87xxx_pa";
                    reg = <0x59>;
                    reset-gpio = <&pio 7 0>; /*带 reset 引脚的配置*/
                    dev_index = < 1 >;
                    status = "okay";
        };
       /* AWINIC AW87XXX Smart K PA End */
    };
```

如需要使用 ESD 检测功能，请在 dtsi 配置中增加配置 esd-enable = “true”;(true 为使能，false 为不使能)

**多 PA 的配置可参考以上双 PA 配置进行增加设备节点即可。**

### 3.1.2 Driver

在内核中 kernel/sound/soc/awinic 添加 aw87xxx 的工程文件：

| | |
|---|---|
| 源文件 | aw87xxx.c, aw_device.c, aw_monitor.c, aw_dsp.c, aw_acf_bin.c, aw_bin_parse.c |
| 头文件 | aw87xxx.h, aw_device.h, aw_monitor.h, aw_dsp.h, aw_acf_bin.h, aw_log.h, aw87xxx_pid_39_reg.h, aw87xxx_pid_59_3x9_reg.h, aw87xxx_pid_59_5x9_reg.h, aw87xxx_pid_5a_reg.h, aw87xxx_pid_18_reg.h, aw87xxx_pid_9b_reg.h |

### 3.1.3 Kconfig && Makefile

在内核中 kernel/sound/soc/mediatek/Kconfig 添加 aw87xxx 的 Kconfig：

```
config SND_SOC_AW87XXX
    tristate "SoC Audio for awinic AW87XXX Smart K PA"
    depends on I2C
help
    This option enables support for AW87XXX Smart K PA.
```

在内核中 kernel/sound/soc/mediatek/Makefile 添加 aw87xxx 的 Makefile：

```
#for AWINIC AW87XXX Smart K PA
obj-$(CONFIG_SND_SOC_AW87XXX) += awinic/aw87xxx.o  awinic/aw_device.o
awinic/aw_monitor.o  awinic/aw_bin_parse.o  awinic/aw_dsp.o
awinic/aw_acf_bin.o
```

### 3.1.4 AW87XXX Config Bin File

在内核 kernel/drivers/base/firmware_class.c 文件中添加 bin 文件的目录，目录由系统决定，一般目录为:/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"
};
```

使用 adb 将 config 文件 aw87xxx_acf.bin 文件 push 到 vendor/firmware 中。

**注：aw87xxx_acf.bin 文件在软件移植包目录下 config 目录中各产品目录下，包含了默认单声道 bin 文件和默认同产品的双 PA 配置，且都包含了默认 Music/Receiver/Off 场景配置和默认 monitor 配置(除过 aw87319 产品，它们是不需要 Off.bin 参数的)。**

**软件移植包下的默认 aw87xxx_acf.bin 合成工具版本为 Awinic_ACF_Tool v0.0.7。**

### 3.1.5 AW87XXX Config update time

在驱动源文件 aw87xxx.h 中设置了可以修改的初始化固件延时加载时间控制宏，默认驱动注册 5000ms 之后加载固件，设置如下：

```
#define AWINIC_CFG_UPDATE_DELAY
#define AWINIC_CFG_UPDATE_DELAY_TIME    (5000)
```

其中宏 AWINIC_CFG_UPDATE_DELAY 定义时则按照 AWINIC_CFG_UPDATE_DELAY_TIME 时间延时之后再加载固件。AWINIC_CFG_UPDATE_DELAY 不定义时则不延时直接加载固件。

该设置可由用户根据客户端文件系统加载时间来修改，以保证能正常发加载固件参数。

## 3.2 Codec Driver

awinic 提供了两种平台端控制 PA 的方案，客户可根据实际需求选择使用。

### 3.2.1 方案一：Kcontrol 控件使能 PA 上下电

#### 1） Kcontrol 注册

使用 Kcontrol 接口进行切换场景和获取实时的低压保护调整参数 vmax 时，需要在平台 codec 代码中添加 aw87xxx_codec 的注册，以 kernel/sound/soc/mediatek/codec/mt6357/mtk-soc-codec-6357.c 为例。

在 mtk-soc-codec-6357.c 中添加 aw87xxx_codec 注册函数的外部引用申明：

```
#ifdef CONFIG_SND_SOC_AW87XXX
extern int aw87xxx_add_codec_controls(void *codec);
#endif /*CONFIG_SND_SOC_AWINIC_AW87XXX*/
```



 

在 mt6357_codec_probe() 函数中添加 aw87xxx_codec 注册函数的调用。

```
#ifdef CONFIG_SND_SOC_AW87XXX
    ret = aw87xxx_add_codec_controls(codec);
    if (ret < 0) {
        pr_err("%s: aw87xxx_add_codec_controls failed, ret= %d\n",
            __func__, ret);
        return ret;
        };
#endif
```

**2） Xml 配置**

awinic 为客户提供了默认的 Music /Receiver /Off 场景，如需其他场景，可通过 aw87xxx_acf.bin 配置增加场景实现，客户可根据需要在 aw87xxx_acf.bin 中合成需要的场景参数，XML 中场景也需要根据具体的 aw87xxx_acf.bin 中包含的场景来配置。

audio_device.xml 是 MTK 音频 path 管理文件，可以对相应的 control 进行 turn on，turn off。在项目对应的 audio_device.xml 文件中添加 AW87XXX 的 kcontrl 控制。

这里以 speaker_output 通路为例：

```
<!--speaker output-->
 <path name="speaker_output" value="turnon">
    <kctl name="aw87xxx_profile_switch_0" value="Music" />
    <kctl name="aw87xxx_profile_switch_1" value="Music" />
    …
 </path>
 <path name="speaker_output" value="turnoff">
    <kctl name="aw87xxx_profile_switch_0" value="Off" />
    <kctl name="aw87xxx_profile_switch_1" value="Off" />
    …
 </path>
```

**3） Kcontrol 接口使用**

**场景切换接口：**

AW87XXX 为客户提供了用于切换场景的 Kcontrol 接口，接口名为 aw87xxx_profile_switch_x（x 为 dev_index，由 dtsi 或者驱动按驱动注册顺序默认配置），即会为每一个注册的设备都创建一个 Kcontrol 接口。

**使用方法：**

| | |
|---|---|
| tinymix aw87xxx_profile_switch_0 | 查看 dev0 成功加载的场景以及当前场景 |
| tinymix aw87xxx_profile_switch_0 Music | 切换 dev0 的场景为 Music |
| tinymix aw87xxx_profile_switch_1 Off | 切换 dev1 的场景为 Off |

**低压保护接口：**

AW87XXX 为客户提供了可用于在 no_dsp 或 with_dsp 时都可以进行获取实时低压保护调整参数的 Kcontrol 接口 aw87xxx_vmax_get_x（x 为每个 PA 的 dev_index 编号）。

使用方法：

tinymix aw87xxx_vmax_get_0　　获取 dev0 的实时电量保护调整参数

举例说明：

```
k39tv1_bsp:/ # tinymix aw87xxx_vmax_get_0
aw87xxx_vmax_get_0: -7272660 (dsrange -2147483648->0)
k39tv1_bsp:/ # tinymix aw87xxx_vmax_get_1
aw87xxx_vmax_get_1: -7272660 (dsrange -2147483648->0)
```

 

```
[name:aw_monitor&][1-0058]aw_monitor_get_battery_capacity: The percentage is 1
[name:aw_monitor&][1-0058]aw_monitor_no_dsp_get_vmax: get_battery_capacity is[1]
[name:aw_monitor&][1-0058]aw_search_vmax_from_table: read setting vmax=0xff91072c, step[0]: vbat_min=0,vbat_max=40
[name:aw87xxx&][1-0058]aw87xxxx_vmax_get: get vmax = [0xff91072c]
[name:aw_monitor&][1-0059]aw_monitor_get_battery_capacity: The percentage is 1
[name:aw_monitor&][1-0059]aw_monitor_no_dsp_get_vmax: get_battery_capacity is[1]
[name:aw_monitor&][1-0059]aw_search_vmax_from_table: read setting vmax=0xff91072c, step[0]: vbat_min=0,vbat_max=40
[name:aw87xxx&][1-0059]aw87xxxx_vmax_get: get vmax = [0xff91072c]
```

### 3.2.2 方案二：平台控件中添加 aw87xxx 场景加载

1） **4G** 平台移植方案：

以 kernel/sound/soc/mediatek/codec/mt6357/mtk-soc-codec-6357.c 为例，添加 aw87xxx 场景加载的函数调用。

在 mtk-soc-codec-6357.c 中添加 aw87xxx 场景切换外部函数声明以及变量定义：

```
@@ -80,6 +80,17 @@ static void setDlMtkifSrc(bool enable);
 #ifndef ANALOG_HPTRIM
 static int SetDcCompenSation(bool enable);
 #endif
+#ifdef CONFIG_SND_SOC_AW87XXX
+extern int aw87xxx_set_profile(int dev_index, char *profile);
+
+static char *aw_profile[] = {"Music", "Off"};
+enum aw87xxx_dev_index {
+       AW_DEV_0 = 0,
+};
+#endif
 static void Voice_Amp_Change(bool enable);
```

注：awinic 为客户提供了 Music/Receiver/Off 场景，如需其他场景，可通过 aw87xxx_acf.bin 配置增加场景实现。以上集成中 aw_profile 数组变量中的场景名字符串需要 awinic_ACF_tool 中配置的场景名保持一致，默认参数配置可以切换的场景为 Music/Receiver/Off。

在指定场景控制函数中添加 aw87xxx 场景加载函数：

```
@@ -3296,6 +3347,9 @@ static void Speaker_Amp_Change(bool enable)
        Ana_Set_Reg(AUDDEC_ANA_CON6, 0x0201, 0xffff);
        /* Switch LOL MUX to audio DAC */
        Ana_Set_Reg(AUDDEC_ANA_CON4, 0x011b, 0xffff);
+#ifdef CONFIG_SND_SOC_AW87XXX
+       aw87xxx_set_profile(AW_DEV_0, aw_profile[0]);
+#endif
        /* disable Pull-down HPL/R to AVSS28_AUD */
        if (mIsNeedPullDown)
                hp_pull_down(false);
@@ -3333,6 +3387,10 @@ static void Speaker_Amp_Change(bool enable)
            Ana_Set_Reg(AUDDEC_ANA_CON12, 0x0, 0x1055);
            /* Disable NCP */
            Ana_Set_Reg(AUDNCP_CLKDIV_CON3, 0x1, 0x1);
+
+#ifdef CONFIG_SND_SOC_AW87XXX
+       aw87xxx_set_profile(AW_DEV_0, aw_profile[1]);
+#endif
            TurnOffDacPower();
        }
    }
```

 

注：以上添加函数调用以单 PA 单场景为例，如需要实现多个 PA 或多个场景控制，请参考以上代码添加对应变量或函数调用。

2） **5G** 平台 **widget** 控制 **PA** 添加方案

```
@@ -17,6 +17,13 @@
 #include "../../codecs/mt6359.h"
 #include "../common/mtk-sp-spk-amp.h"

+#ifdef CONFIG_SND_SOC_AW87XXX
+extern int aw87xxx_set_profile(int dev_index, char *profile);
+static char *aw_profile[] = {"Music", "Off"};
+enum aw87xxx_dev_index {
+     AW_DEV_0 = 0,
+};
+#endif

 /*
  * if need additional control for the ext spk amp that is connected
@@ -92,9 +99,25 @@ static int mt6853_mt6359_spk_amp_event(s
    switch (event) {
    case SND_SOC_DAPM_POST_PMU:
        /* spk amp on control */
+#ifdef CONFIG_SND_SOC_AW87XXX
+       ret = aw87xxx_set_profile(AW_DEV_0, aw_profile[0]);
+       if (ret < 0) {
+           pr_err("[Awinic] %s: set profile[%s] failed",
+                  __func__, aw_profile[0]);
+           return ret;
+       }
+#endif
        break;
    case SND_SOC_DAPM_PRE_PMD:
        /* spk amp off control */
+#ifdef CONFIG_SND_SOC_AW87XXX
+       ret = aw87xxx_set_profile(AW_DEV_0, aw_profile[1]);
+       if (ret < 0) {
+           pr_err("[Awinic] %s: set profile[%s] failed",
+                  __func__, aw_profile[1]);
+           return ret;
+       }
+#endif
        break;
    default:
        break;
```

注：awinic 为客户提供了 Music/Receiver/Off 场景，如需其他场景，可通过 aw87xxx_acf.bin 配置增加场景实现。以上集成中 aw_profile 数组变量中的场景名字符串需要 awinic_ACF_tool 中配置的场景名保持一致，默认参数配置可以切换的场景为 Music/Receiver/Off。

以上添加函数调用以单 PA 单场景为例，如需要实现多个 PA 或多个场景控制，请参考以上代码添加对应变量或函数调用。

# 4. Debug Interface

## 4.1 attributes nodes

AW87XXX Driver 会创建不同设备节点，路径是 sys/bus/i2c/driver/aw87xxx_pa/*-00xx，

在每个设备文件下各创建 reg/profile/hwen/esd_enable 4 个设备节点，其中*为 i2c bus number，xx 为 i2c address。

**1） reg**

| 节点名字 | **reg** |
|---|---|
| 功能描述 | 用于读写 aw87xxx 的所有寄存器 |
| 使用方法 | 读寄存器值：cat reg<br>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）<br>echo 0x01 0x07 > reg （向 0x01 寄存器写入 0x07 的值） |

**2） profile**

| 节点名字 | **profile** |
|---|---|
| 功能描述 | 用于 aw87xxx 注册的 PA 切换场景 |
| 使用方法 | 查看当前场景和可切换的场景：cat profile<br>设置场景：echo 场景名 > profile |
| 参考例程 | cat reg （查看当前场景和可切换的场景）<br>echo “Music” > profile （加载 Music 场景）<br>echo “Off” > profile （加载 Off 场景（PA 下电）） |

**1） hwen**

| 节点名字 | **hwen** |
|---|---|
| 功能描述 | 用于控制 aw87xxx 的硬件开关 |
| 使用方法 | cat hwen （获取 aw87xxx 硬件硬件状态）<br>echo 1 > hwen （aw87xxx 硬件使能）<br>echo 0 > hwen （aw87xxx 硬件关闭） |

**2） esd_enable**

| 节点名字 | **esd_enable** |
|---|---|
| 功能描述 | 用于调试开关 esd 检测使能 |
| 使用方法 | 读：cat esd_enable<br>写：echo is_enable > esd_enable （字符串） |
| 参考例程 | cat esd_enable （获取所有可读寄存器上的值）<br>echo true > esd_enable （打开 esd 检测功能）<br>echo false > esd_enable （关闭 esd 检测功能） |

# 5. 低电量保护算法

**注：以下环节仅针对于需要低电量保护算法的客户，如果不需要可不关注：**

低电量保护算法根据平台是否含有 open dsp，设计了两种方案: 1. No dsp 低电量保护算法；2.with dsp 低电量保护算法。

## 5.1 config 配置

对于 no_dsp 和 with_dsp 都需要配置 config，参数配置请参考《aw87xxx_monitor_bin_guide.pdf》，区别在于对于 no_dsp 的 config 配置的 monitor_switch/monitor_count/monitor_time 无效，可根据说明配置。

## 5.2 No dsp 低电量保护算法

若平台不含有 open dsp， 请参考算法移植文档《awinic_skt_mtk_porting.pdf》，进行移植，通过注册的 Kcontrol 接口 aw87xxx_vmax_get_x（x 为 PA 的 dev_index）获取实时的低电量保护调整值。

注意：No dsp 低电量保护算法兼容 v2.0.0 以前版本，可通过 vmax 节点获取实时的低电量调整值，因此在使用 v2.0.0 版本保护算法加载之前应该 vmax 节点对用户有可读权限。

### 5.2.1 Debug Interface

AW87XXX Driver 对于 no_dsp 的低电量保护会在设备文件下创建 vmax/vbat 两个设备节点，路径是 sys/bus/i2c/driver/aw87xxx_pa/*-00xx，其中*为 i2c bus number，xx 为 i2c address。

**1）vmax**

| 节点名字 | **vmax** |
|---|---|
| 功能描述 | 用于获取实时电量在 monitor_bin 配置中查找的 vmax 值 |
| 使用方法 | cat vmax （获取当前 vmax 的值） |

**1）vbat**

| 节点名字 | **vbat** |
|---|---|
| 功能描述 | 用于调试设置当前电量和获取系统实时的电量值 |
| 使用方法 | cat vbat （获取实时电量值）<br>echo capacity > vbat （设置 vbat 电量值）<br>echo 0 > vbat （取消之前输入的调试电量） |
| 参考例程 | echo 50 > vbat （设置当前电量为 50%） |

## 5.3 With dsp 低电量保护算法

若平台带有 open dsp，请在 aw_dsp.h 中的增加 MTK 平台 dsp 相关的宏定义。

```
/*#define AW_MTK_OPEN_DSP_PLATFORM*/
```

### 5.3.1 Debug Interface

AW87XXX Driver 会在带 dsp 时在每个设备路径 sys/bus/i2c/driver/aw87xxx_pa/*-00xx 下创建 vmax/vbat/monitor_switch/monitor_count/monitor_time/rx 6 个设备节点，其中*为 i2c bus number，xx 为 i2c address。

**1） vmax**

| 节点名字 | **vmax** |
|---|---|
| 功能描述 | 用于向 dsp 设置 vmax 和获取 dsp 当前的 vmax 值（获取的 vmax 为算法中计算出的 vmax 值，与设置的值不相同） |
| 使用方法 | cat vmax （获取当前 vmax 的值） |

| | | |
|---|---|---|
| | echo N > vmax | （发送计算后的 vmax 值） |
| 参考例程 | echo 0xfff95f7e > vmax | （将 0xfff95f7e 值发送给 vmax） |

**2） vbat**

| | | |
|---|---|---|
| 节点名字 | **vbat** | |
| 功能描述 | 用于调试设置当前电量和获取实时的电量值 | |
| 使用方法 | cat vbat<br>echo capacity > vbat<br>echo 0 > vbat | （获取实时电量值）<br>（设置 vbat 电量值）<br>（取消之前输入的调试电量） |
| 参考例程 | echo 50 > vbat | （设置当前电量为 50%） |

**3） monitor_switch**

| | | |
|---|---|---|
| 节点名字 | **monitor_switch** | |
| 功能描述 | 用于设置 aw87xxx 的带 dsp 自动保护功能使能 | |
| 使用方法 | cat monitor_switch<br>echo 0 > monitor_switch<br>echo 1 > monitor_switch | （获取当前电量的保护状态）<br>（aw87xxx 关闭自动保护）<br>（aw87xxx 开启自动保护） |

**4） monitor_count**

| | | |
|---|---|---|
| 节点名字 | **monitor_count** | |
| 功能描述 | 用于设置 aw87xxx 的带 dsp 自动保护向 dsp 设置一次 vmax 前获取电量求均的次数 | |
| 使用方法 | cat monitor_count<br>echo 5 > monitor_count<br>echo 10 > monitor_count | （获取系统当前电量获取次数）<br>（设置获取电量求均次数为 5）<br>（设置获取电量求均次数为 10） |

**5） monitor_time**

| | | |
|---|---|---|
| 节点名字 | **monitor_time** | |
| 功能描述 | 用于设置获取电量的时间间隔 | |
| 使用方法 | cat monitor_time<br>echo N > monitor_time | （获取系统当前获取电量的时间间隔（ms））<br>（设置 N（ms）时间间隔后获取一次电量） |

**6） rx**

| | | |
|---|---|---|
| 节点名字 | **rx** | |
| 功能描述 | 用于设置和获取 dsp rx 模块的状态 | |
| 使用方法 | cat rx<br>echo 1 > rx<br>echo 0 > rx | （获取 dsp 的 rx 模块的状态）<br>（设置 dsp 的 rx 模块使能）<br>（设置 dsp 的 rx 模块不使能） |



 