awinic SKT 算法集成（MTK）

时间： 2020.10.09
版本： v1.0

修订记录：

版本	日期	说明
V1.0	2020 年 10 月	

目录

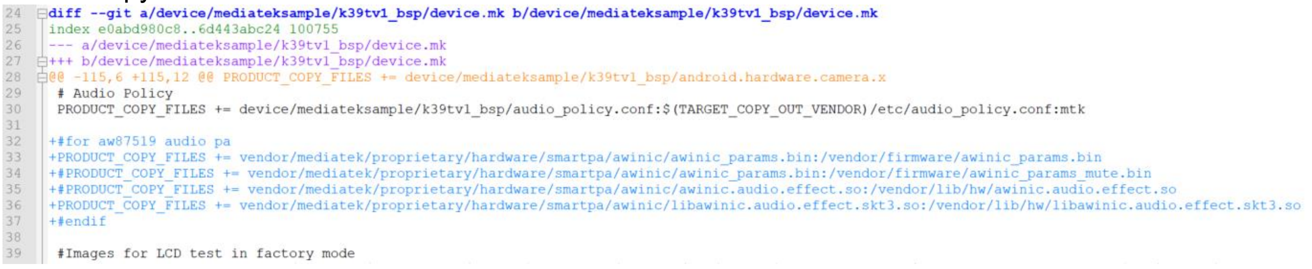
1. 简介

本文档介绍了 MTK AP 侧集成 AWINIC SKT 算法的操作方法。为用户集成 AWINIC 算法提供指导。

2. 算法集成

1）将算法动态库（libawinic.audio.effect.so）以及参数文件（awinic_params.bin）copy 到用户的客制化目录，这里以 vendor/mediatek/proprietary/hardware/smartpa/awinic/为例

2）在对应项目的 device.mk 中添加 copy 操作，将文件 copy 到指定的目录，如果使用的是 64 位的系统这需要 copy64 位的算法库（MTK 基本上都为 32bit 库文件）

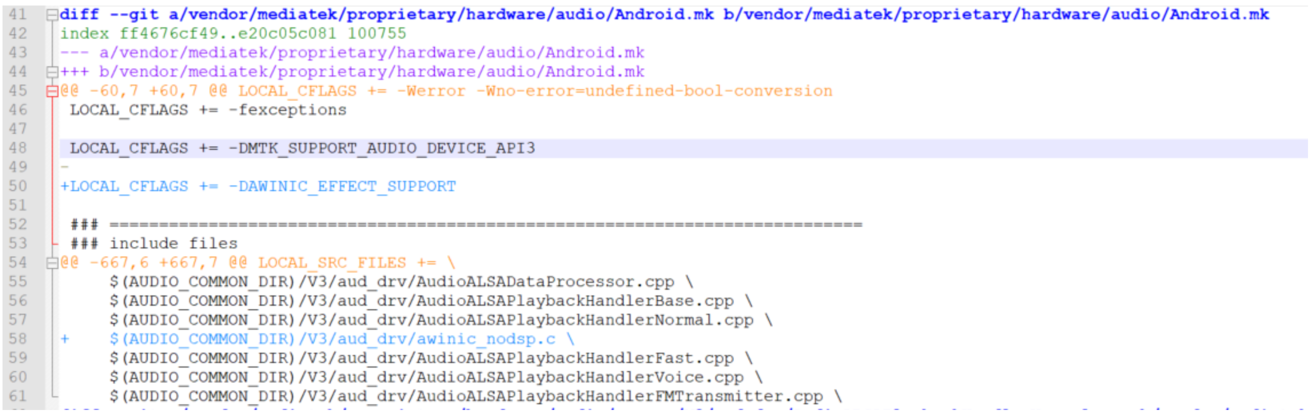


```
diff --git a/device/mediateksample/k39tv1_bsp/device.mk b/device/mediateksample/k39tv1_bsp/device.mk
index e0abd980c8..6d443abc24 100755
--- a/device/mediateksample/k39tv1_bsp/device.mk
+++ b/device/mediateksample/k39tv1_bsp/device.mk
@@ -115,6 +115,12 @@ PRODUCT_COPY_FILES += device/mediateksample/k39tv1_bsp/android.hardware.camera.x
 # Audio Policy
 PRODUCT_COPY_FILES += device/mediateksample/k39tv1_bsp/audio_policy.conf:$(TARGET_COPY_OUT_VENDOR)/etc/audio_policy.conf:mtk

+#for aw87519 audio pa
+PRODUCT_COPY_FILES += vendor/mediatek/proprietary/hardware/smartpa/awinic/awinic_params.bin:/vendor/firmware/awinic_params.bin
+#PRODUCT_COPY_FILES += vendor/mediatek/proprietary/hardware/smartpa/awinic/awinic_params.bin:/vendor/firmware/awinic_params_mute.bin
+#PRODUCT_COPY_FILES += vendor/mediatek/proprietary/hardware/smartpa/awinic/awinic.audio.effect.so:/vendor/lib/hw/awinic.audio.effect.so
+PRODUCT_COPY_FILES += vendor/mediatek/proprietary/hardware/smartpa/awinic/libawinic.audio.effect.skt3.so:/vendor/lib/hw/libawinic.audio.effect.skt3.so
+#endif

 #Images for LCD test in factory mode
```

3）修改 vendor/mediatek/proprietary/hardware/audio/Android.mk
添加宏控:

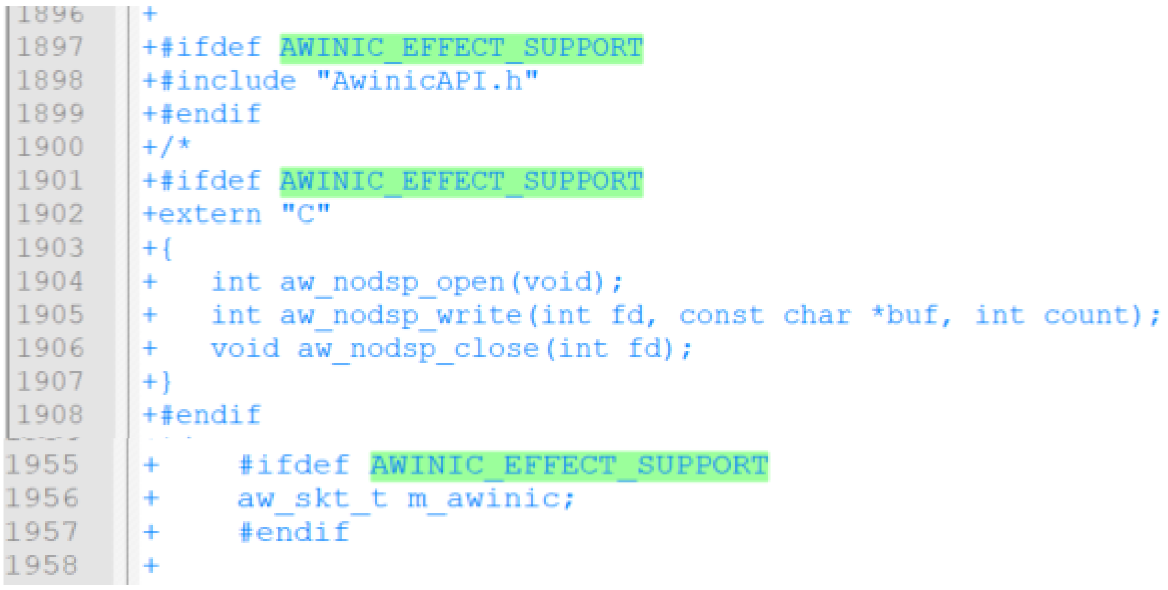


```
diff --git a/vendor/mediatek/proprietary/hardware/audio/Android.mk b/vendor/mediatek/proprietary/hardware/audio/Android.mk
index ff4676cf49..e20c05c081 100755
--- a/vendor/mediatek/proprietary/hardware/audio/Android.mk
+++ b/vendor/mediatek/proprietary/hardware/audio/Android.mk
@@ -60,7 +60,7 @@ LOCAL_CFLAGS += -Werror -Wno-error=undefined-bool-conversion
 LOCAL_CFLAGS += -fexceptions

 LOCAL_CFLAGS += -DMTK_SUPPORT_AUDIO_DEVICE_API3
-
+LOCAL_CFLAGS += -DAWINIC_EFFECT_SUPPORT

 ### ============================================================================
 ### include files
@@ -667,6 +667,7 @@ LOCAL_SRC_FILES += \
        $(AUDIO_COMMON_DIR)/V3/aud_drv/AudioALSADataProcessor.cpp \
        $(AUDIO_COMMON_DIR)/V3/aud_drv/AudioALSAPlaybackHandlerBase.cpp \
        $(AUDIO_COMMON_DIR)/V3/aud_drv/AudioALSAPlaybackHandlerNormal.cpp \
+       $(AUDIO_COMMON_DIR)/V3/aud_drv/awinic_nodsp.c \
        $(AUDIO_COMMON_DIR)/V3/aud_drv/AudioALSAPlaybackHandlerFast.cpp \
        $(AUDIO_COMMON_DIR)/V3/aud_drv/AudioALSAPlaybackHandlerVoice.cpp \
        $(AUDIO_COMMON_DIR)/V3/aud_drv/AudioALSAPlaybackHandlerFMTransmitter.cpp \
```

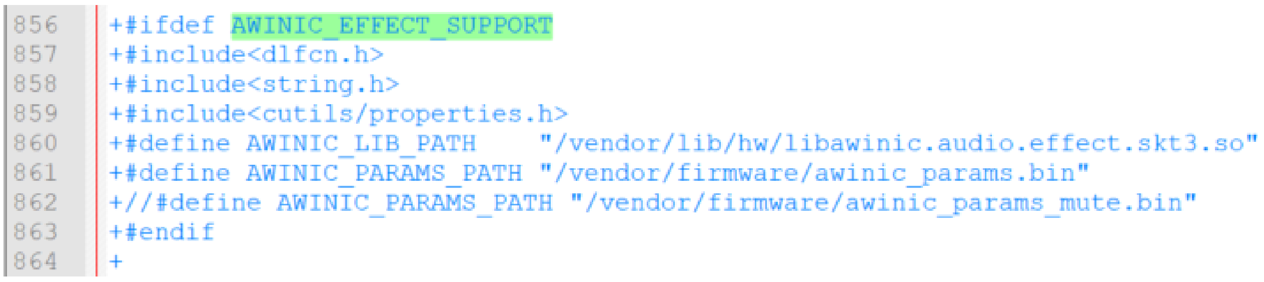
4）将 AwinicAPI.h 复制到 vendor/mediatek/proprietary/hardware//audio/common/V3/include/中
修改
vendor/mediatek/proprietary/hardware/audio/common/V3/include/AudioALSAPlaybackHandlerNormal.h
文件，添加 awinic 算法变量



```
+
+#ifdef AWINIC_EFFECT_SUPPORT
+#include "AwinicAPI.h"
+#endif
+/*
+#ifdef AWINIC_EFFECT_SUPPORT
+extern "C"
+{
+    int aw_nodsp_open(void);
+    int aw_nodsp_write(int fd, const char *buf, int count);
+    void aw_nodsp_close(int fd);
+}
+#endif
+     #ifdef AWINIC_EFFECT_SUPPORT
+     aw_skt_t m_awinic;
+     #endif
+
```

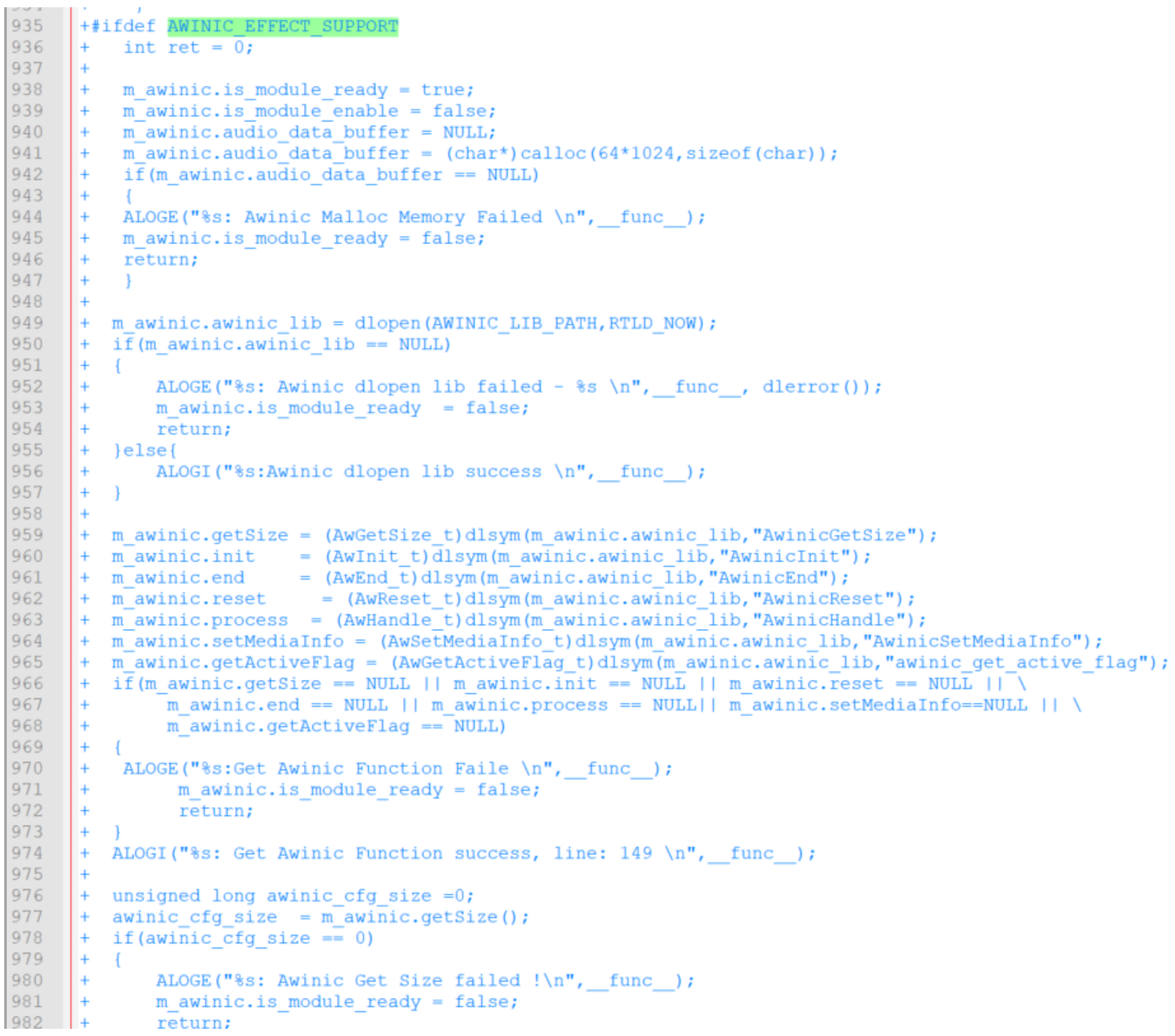
5）修改 vendor/mediatek/proprietary/hardware/audio/common/V3/aud_drv/AudioALSAPlaybackHandlerNormal.cpp 文件，添加 awinic 算法调用

添加头文件：



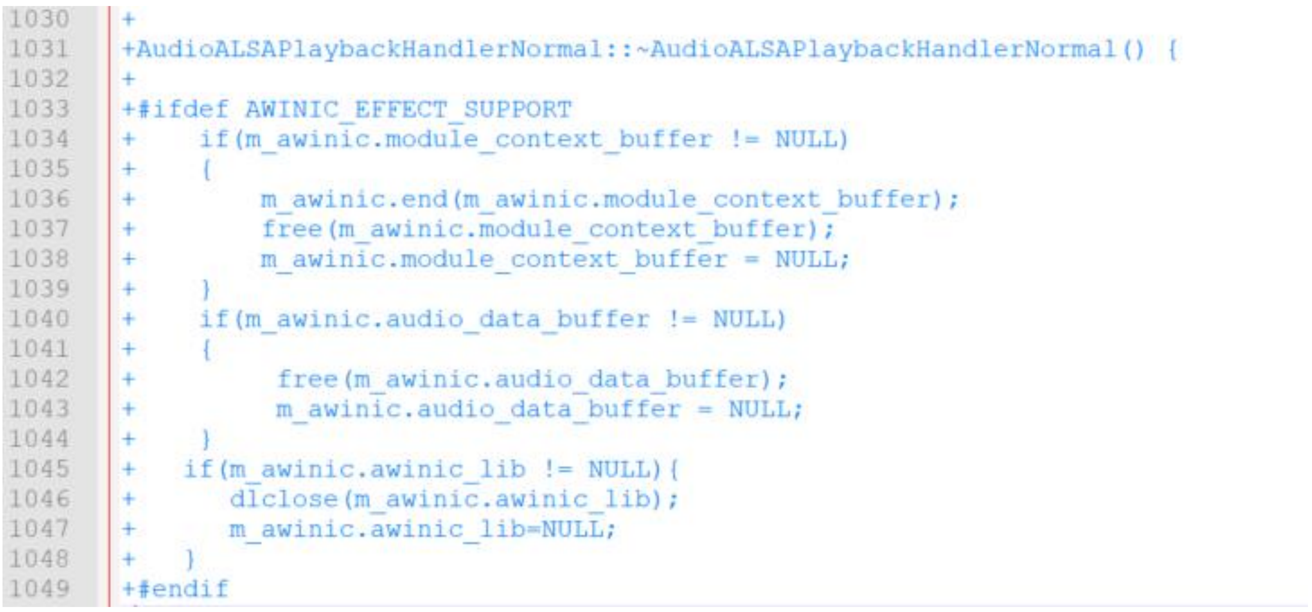
```
+#ifdef AWINIC_EFFECT_SUPPORT
+#include<dlfcn.h>
+#include<string.h>
+#include<cutils/properties.h>
+#define AWINIC_LIB_PATH    "/vendor/lib/hw/libawinic.audio.effect.skt3.so"
+#define AWINIC_PARAMS_PATH "/vendor/firmware/awinic_params.bin"
+//#define AWINIC_PARAMS_PATH "/vendor/firmware/awinic_params_mute.bin"
+#endif
+
```

在初始化函数（AudioALSAPlaybackHandlerNormal）添加初始化代码，详细的可以参考 patch：



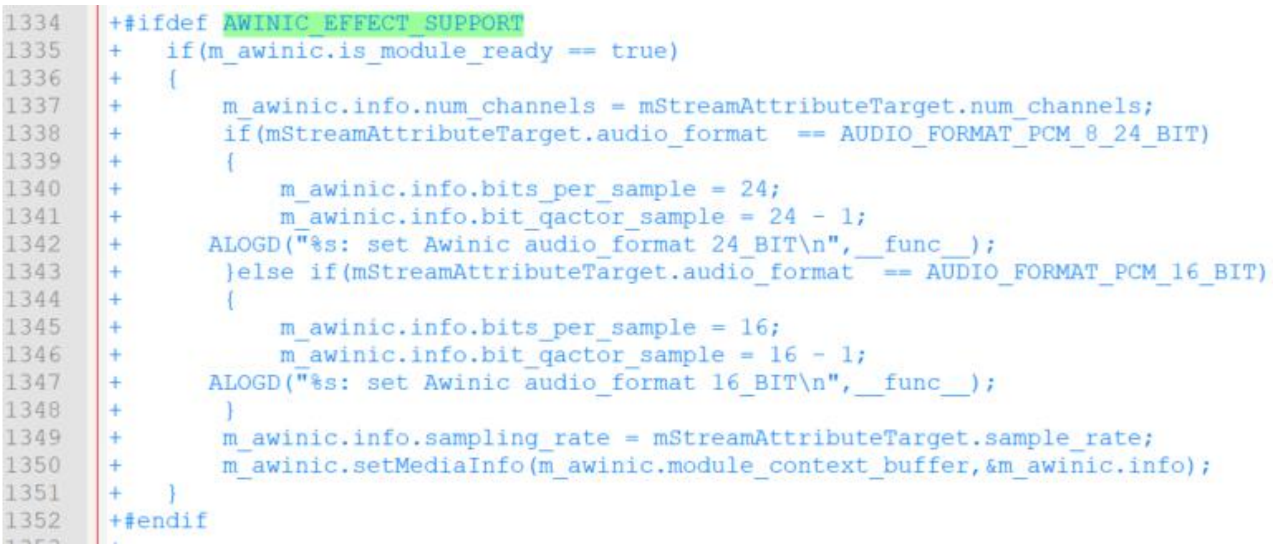
```
+#ifdef AWINIC_EFFECT_SUPPORT
+    int ret = 0;
+
+    m_awinic.is_module_ready = true;
+    m_awinic.is_module_enable = false;
+    m_awinic.audio_data_buffer = NULL;
+    m_awinic.audio_data_buffer = (char*)calloc(64*1024,sizeof(char));
+    if(m_awinic.audio_data_buffer == NULL)
+    {
+    ALOGE("%s: Awinic Malloc Memory Failed \n",__func__);
+    m_awinic.is_module_ready = false;
+    return;
+    }
+
+   m_awinic.awinic_lib = dlopen(AWINIC_LIB_PATH,RTLD_NOW);
+   if(m_awinic.awinic_lib == NULL)
+   {
+       ALOGE("%s: Awinic dlopen lib failed - %s \n",__func__, dlerror());
+       m_awinic.is_module_ready  = false;
+       return;
+   }else{
+       ALOGI("%s:Awinic dlopen lib success \n",__func__);
+   }
+
+   m_awinic.getSize = (AwGetSize_t)dlsym(m_awinic.awinic_lib,"AwinicGetSize");
+   m_awinic.init     = (AwInit_t)dlsym(m_awinic.awinic_lib,"AwinicInit");
+   m_awinic.end      = (AwEnd_t)dlsym(m_awinic.awinic_lib,"AwinicEnd");
+   m_awinic.reset      = (AwReset_t)dlsym(m_awinic.awinic_lib,"AwinicReset");
+   m_awinic.process  = (AwHandle_t)dlsym(m_awinic.awinic_lib,"AwinicHandle");
+   m_awinic.setMediaInfo = (AwSetMediaInfo_t)dlsym(m_awinic.awinic_lib,"AwinicSetMediaInfo");
+   m_awinic.getActiveFlag = (AwGetActiveFlag_t)dlsym(m_awinic.awinic_lib,"awinic_get_active_flag");
+   if(m_awinic.getSize == NULL || m_awinic.init == NULL || m_awinic.reset == NULL || \
+        m_awinic.end == NULL || m_awinic.process == NULL|| m_awinic.setMediaInfo==NULL || \
+        m_awinic.getActiveFlag == NULL)
+   {
+    ALOGE("%s:Get Awinic Function Faile \n",__func__);
+         m_awinic.is_module_ready = false;
+         return;
+   }
+   ALOGI("%s: Get Awinic Function success, line: 149 \n",__func__);
+
+   unsigned long awinic_cfg_size =0;
+   awinic_cfg_size  = m_awinic.getSize();
+   if(awinic_cfg_size == 0)
+   {
+       ALOGE("%s: Awinic Get Size failed !\n",__func__);
+       m_awinic.is_module_ready = false;
+       return;
```

在析构函数添加释放内存代码（~AudioALSAPlaybackHandlerNormal）：



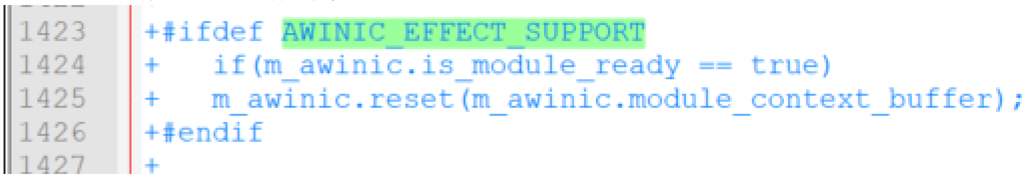
```
+
+AudioALSAPlaybackHandlerNormal::~AudioALSAPlaybackHandlerNormal() {
+
+#ifdef AWINIC_EFFECT_SUPPORT
+     if(m_awinic.module_context_buffer != NULL)
+     {
+         m_awinic.end(m_awinic.module_context_buffer);
+         free(m_awinic.module_context_buffer);
+         m_awinic.module_context_buffer = NULL;
+     }
+     if(m_awinic.audio_data_buffer != NULL)
+     {
+          free(m_awinic.audio_data_buffer);
+          m_awinic.audio_data_buffer = NULL;
+     }
+    if(m_awinic.awinic_lib != NULL){
+       dlclose(m_awinic.awinic_lib);
+       m_awinic.awinic_lib=NULL;
+    }
+#endif
```

在 open 函数添加设置 media 格式信息代码：



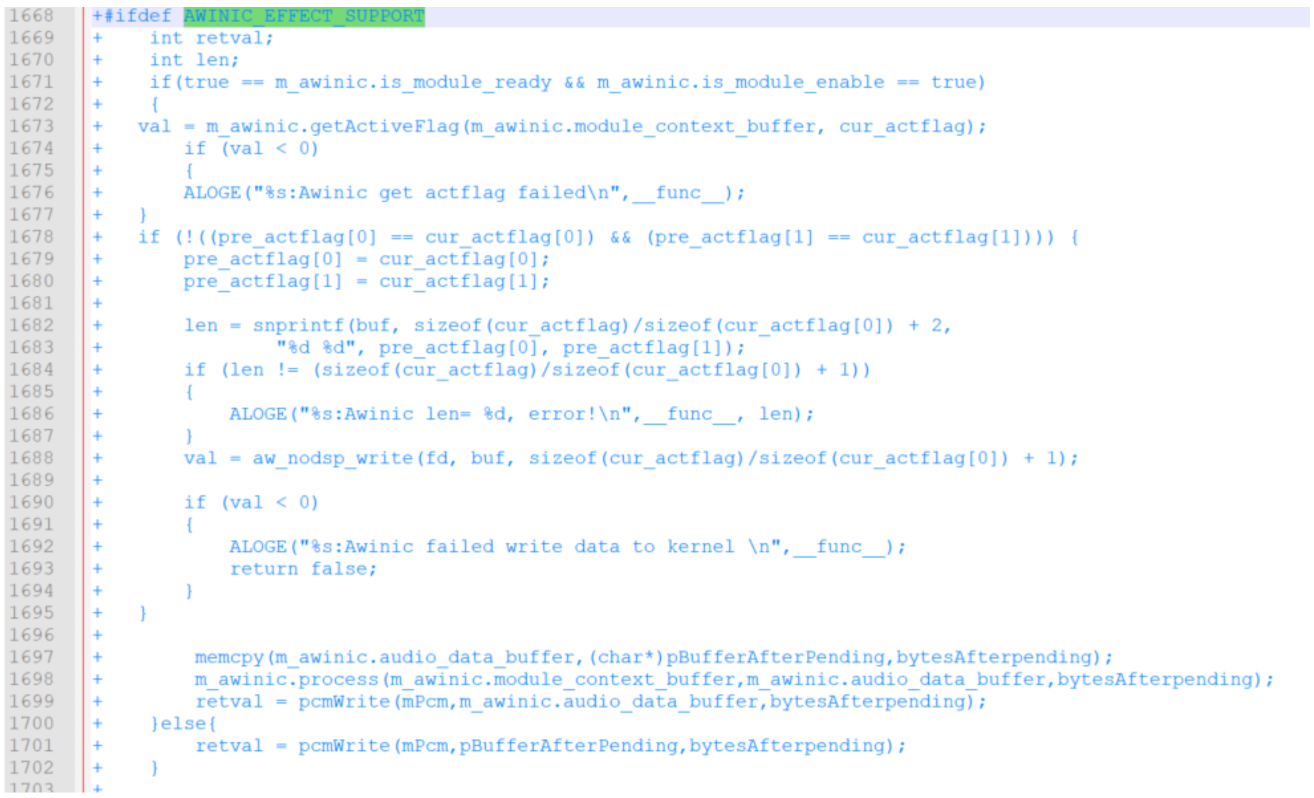
```
+#ifdef AWINIC_EFFECT_SUPPORT
+    if(m_awinic.is_module_ready == true)
+    {
+        m_awinic.info.num_channels = mStreamAttributeTarget.num_channels;
+        if(mStreamAttributeTarget.audio_format   == AUDIO_FORMAT_PCM_8_24_BIT)
+        {
+            m_awinic.info.bits_per_sample = 24;
+            m_awinic.info.bit_qactor_sample = 24 - 1;
+       ALOGD("%s: set Awinic audio_format 24_BIT\n",__func__);
+        }else if(mStreamAttributeTarget.audio_format   == AUDIO_FORMAT_PCM_16_BIT)
+        {
+            m_awinic.info.bits_per_sample = 16;
+            m_awinic.info.bit_qactor_sample = 16 - 1;
+       ALOGD("%s: set Awinic audio_format 16_BIT\n",__func__);
+        }
+        m_awinic.info.sampling_rate = mStreamAttributeTarget.sample_rate;
+        m_awinic.setMediaInfo(m_awinic.module_context_buffer,&m_awinic.info);
+    }
+#endif
```

在 close 函数中添加清除代码：



```
+#ifdef AWINIC_EFFECT_SUPPORT
+    if(m_awinic.is_module_ready == true)
+    m_awinic.reset(m_awinic.module_context_buffer);
+#endif
+
```

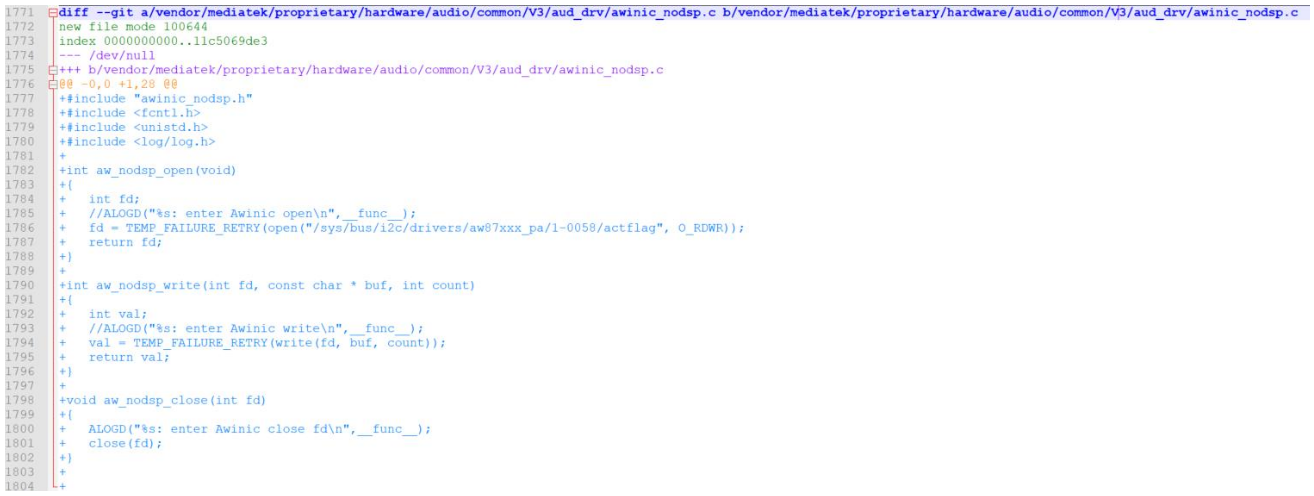
在 write 函数中调用算法：



```
+#ifdef AWINIC_EFFECT_SUPPORT
+    int retval;
+    int len;
+    if(true == m_awinic.is_module_ready && m_awinic.is_module_enable == true)
+    {
+   val = m_awinic.getActiveFlag(m_awinic.module_context_buffer, cur_actflag);
+       if (val < 0)
+       {
+       ALOGE("%s:Awinic get actflag failed\n",__func__);
+   }
+   if (!((pre_actflag[0] == cur_actflag[0]) && (pre_actflag[1] == cur_actflag[1]))) {
+       pre_actflag[0] = cur_actflag[0];
+       pre_actflag[1] = cur_actflag[1];
+
+       len = snprintf(buf, sizeof(cur_actflag)/sizeof(cur_actflag[0]) + 2,
+               "%d %d", pre_actflag[0], pre_actflag[1]);
+       if (len != (sizeof(cur_actflag)/sizeof(cur_actflag[0]) + 1))
+       {
+           ALOGE("%s:Awinic len= %d, error!\n",__func__, len);
+       }
+       val = aw_nodsp_write(fd, buf, sizeof(cur_actflag)/sizeof(cur_actflag[0]) + 1);
+
+       if (val < 0)
+       {
+           ALOGE("%s:Awinic failed write data to kernel \n",__func__);
+           return false;
+       }
+   }
+
+        memcpy(m_awinic.audio_data_buffer,(char*)pBufferAfterPending,bytesAfterpending);
+        m_awinic.process(m_awinic.module_context_buffer,m_awinic.audio_data_buffer,bytesAfterpending);
+        retval = pcmWrite(mPcm,m_awinic.audio_data_buffer,bytesAfterpending);
+    }else{
+        retval = pcmWrite(mPcm,pBufferAfterPending,bytesAfterpending);
+    }
+
```

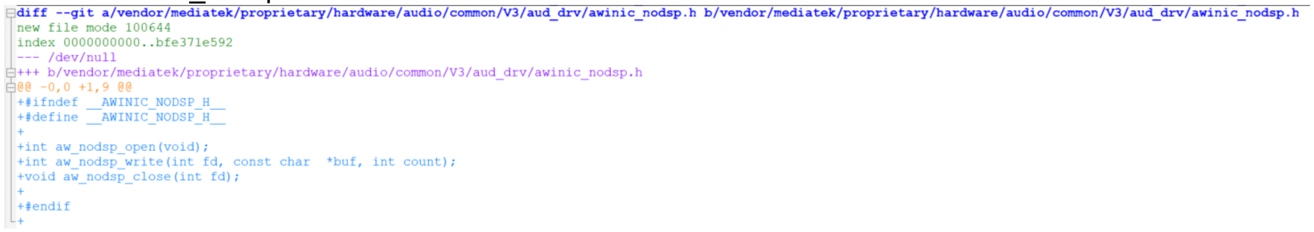
具体代码可以参考附件的 awinic_effect.patch 文件。

6）加入 awinic_nodsp.c



```
diff --git a/vendor/mediatek/proprietary/hardware/audio/common/V3/aud_drv/awinic_nodsp.c b/vendor/mediatek/proprietary/hardware/audio/common/V3/aud_drv/awinic_nodsp.c
new file mode 100644
index 0000000000..11c5069de3
--- /dev/null
+++ b/vendor/mediatek/proprietary/hardware/audio/common/V3/aud_drv/awinic_nodsp.c
@@ -0,0 +1,28 @@
+#include "awinic_nodsp.h"
+#include <fcntl.h>
+#include <unistd.h>
+#include <log/log.h>
+
+int aw_nodsp_open(void)
+{
+   int fd;
+   //ALOGD("%s: enter Awinic open\n",__func__);
+   fd = TEMP_FAILURE_RETRY(open("/sys/bus/i2c/drivers/aw87xxx_pa/1-0058/actflag", O_RDWR));
+   return fd;
+}
+
+int aw_nodsp_write(int fd, const char * buf, int count)
+{
+   int val;
+   //ALOGD("%s: enter Awinic write\n",__func__);
+   val = TEMP_FAILURE_RETRY(write(fd, buf, count));
+   return val;
+}
+
+void aw_nodsp_close(int fd)
+{
+   ALOGD("%s: enter Awinic close fd\n",__func__);
+   close(fd);
+}
+
+
```

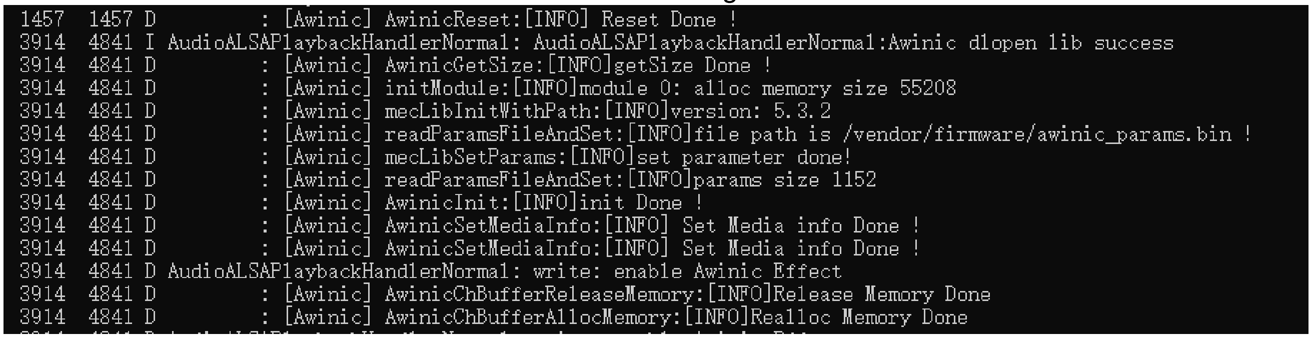
7）加入 awinic_nodsp.h



```
diff --git a/vendor/mediatek/proprietary/hardware/audio/common/V3/aud_drv/awinic_nodsp.h b/vendor/mediatek/proprietary/hardware/audio/common/V3/aud_drv/awinic_nodsp.h
new file mode 100644
index 0000000000..bfe371e592
--- /dev/null
+++ b/vendor/mediatek/proprietary/hardware/audio/common/V3/aud_drv/awinic_nodsp.h
@@ -0,0 +1,9 @@
+#ifndef __AWINIC_NODSP_H__
+#define __AWINIC_NODSP_H__
+
+int aw_nodsp_open(void);
+int aw_nodsp_write(int fd, const char  *buf, int count);
+void aw_nodsp_close(int fd);
+
+#endif
+
```

3 有效性验证

1) 首先抓取 logcat log 看一下是否正常调用算法
adb logcat –v time > awinic.log
然后使用播放器播放音乐，大概 5s 后停止，打开 awinic.log 文件，搜索 Awinic 关键字：



```
1457  1457 D          : [Awinic] AwinicReset:[INFO] Reset Done !
3914  4841 I AudioALSAPlaybackHandlerNormal: AudioALSAPlaybackHandlerNormal:Awinic dlopen lib success
3914  4841 D          : [Awinic] AwinicGetSize:[INFO]getSize Done !
3914  4841 D          : [Awinic] initModule:[INFO]module 0: alloc memory size 55208
3914  4841 D          : [Awinic] mecLibInitWithPath:[INFO]version: 5.3.2
3914  4841 D          : [Awinic] readParamsFileAndSet:[INFO]file path is /vendor/firmware/awinic_params.bin !
3914  4841 D          : [Awinic] mecLibSetParams:[INFO]set parameter done!
3914  4841 D          : [Awinic] readParamsFileAndSet:[INFO]params size 1152
3914  4841 D          : [Awinic] AwinicInit:[INFO]init Done !
3914  4841 D          : [Awinic] AwinicSetMediaInfo:[INFO] Set Media info Done !
3914  4841 D          : [Awinic] AwinicSetMediaInfo:[INFO] Set Media info Done !
3914  4841 D AudioALSAPlaybackHandlerNormal: write: enable Awinic Effect
3914  4841 D          : [Awinic] AwinicChBufferReleaseMemory:[INFO]Release Memory Done
3914  4841 D          : [Awinic] AwinicChBufferAllocMemory:[INFO]Realloc Memory Done
```

正常情况会出现如上 log，如果有报错则需要根据错误确认问题。

2） 确认算法是否运行
使用附件 awinic_params_mute.bin 文件替换手机中的 awinic_params.bin 文件
然后再去播放音乐会出现静音的现象，这样说明调用成功。
如需恢复就将附件中的 awinic_params.bin 重新 push 到手机即可。

4 总结

本文档主要说明了 AWINIC SKT 算法在 MTK AP 集成的具体方法与步骤，用于指导用户进行算法的集成工作。