麦克风阵列音频采集程序集成说明文档

一、麦克风阵列mic_array的Android HAL层实现

1.1 背景说明

SDK中提供了支持麦克风阵列内置录音机，为了使内置录音机能够正常工作，厂商需要对麦克风阵列的打开及数据读取进行Android HAL层的实现，并将SDK中提供的音频读取程序集成到系统中。

此部分的集成包括两个方面：

- 麦克风阵列的Android HAL层实现：
  - 我们提供了`mic_array.h`和`mic_array.c`的参考实现，厂商在集成时需要根据自己硬件麦克风阵列的打开及读取方式，实现对应的接口，基于参考代码，通常只需要根据声卡驱动的参数进行一些配置即可。
- 音频采集程序的集成：
  - 我们提供了一个名为`lothalproc`的可执行程序用于对`mic_array`的接口调用以及向Rokid声学前端SDK的音频数据进行分发。

1.2 mic_array 要实现的功能

- mic_array模块要实现麦克风阵列的打开/读取数据功能,然后 `lothalproc` 进程会根据标准的 `android hw_module_t` 的 `open/start_stream/read_stream` 等方法读取mic阵列的数据,送给声学前端SDK,进行数据处理.

1.3 mic_array要实现的接口

```
int (*get_stream_buff_size) (struct mic_array_device_t *dev);
int (*start_stream) (struct mic_array_device_t *dev);
int (*stop_stream) (struct mic_array_device_t *dev);
int (*finish_stream) (struct mic_array_device_t * dev);
int (*resume_stream) (struct mic_array_device_t *dev);
int (*read_stream) (struct mic_array_device_t *dev, char *buff, unsigned int
 frame_cnt);
int (*config_stream) (struct mic_array_device_t *dev, int cmd, char *cmd_buf
f);
```

- 如上接口均为Android Audio HAL层标准命名接口(可参考 `mic_array.h`) 定义)，需要厂商根据自己硬件麦克风阵列的打开以及读取方式,实现对应接口

1.4 mic_array 几个主要的参数

在 `mic_array.c` 的参考实现中，我们给出了几个重要的参数配置，如下：

```
#define MIC_CHANNEL 6
#define MIC_SAMPLE_RATE 48000

#define PCM_CARD 0
#define PCM_DEVICE 0

static struct pcm_config pcm_config_in = {
    .channels = MIC_CHANNEL,
    .rate = MIC_SAMPLE_RATE,
    .period_size = 480,
    .period_count = 8,
    .format = PCM_FORMAT_S24_LE,
};
```

以上相关的几个参数说明如下：

- `channels` - 为麦克风声道数，表示录音时有几路输入。**注意：**这里的麦克风通道数包含了

AEC参考信号的通道（如果有），假设物理麦克风共**4**麦，AEC参考信号通道有**2**路，则麦克风通道数为**6**。

- `rate` - 为麦克风录音时的采样率，即每秒的采样次数，针对帧而言,目前支持最少16000
- `period_size` - 每次硬件中断处理音频数据的帧数
- `period_count` - 处理完一个buffer数据所需的硬件中断次数
- `format` - 样本长度，音频数据最基本的单位,比如PCM_FORMAT_S32_LE/PCM_FORMAT_S24_LE

其中，`rate`，`period_size`，`period_count`和`format`可以通过声卡的配置获得。

通常Android系统中的声卡驱动，会挂载在`/proc/asound/`节点下，`mic_array`中的声卡参数配置，在获取了`adb root`的权限后，可通过以下方法获得：

- 首先查看声卡信息：

```
$ cat /proc/asound/cards

0 [msm8974taikomtp]: msm8974-taiko-m - msm8974-taiko-mtp-snd-card
                     msm8974-taiko-mtp-snd-card
```

根据以上信息，可知：

```
声卡序号 - PCM_CARD 的值为 0
声卡设备ID 为 msm8974taikomtp
声卡设备名称 为 msm8974-taiko-m
```

- 查看声卡的硬件配置：

```
$ cat /proc/asound/card0/pcm0c/sub0/hw_params

access: RW_INTERLEAVED
format: S24_LE
subformat: STD
channels: 8
rate: 48000 (48000/1)
period_size: 480
buffer_size: 3840
```

由以上信息可知：

- `rate` 为 `48000`

- `period_size` 为 `480`
- `period_count` 通过 `buffer_size/period_size` 可得到，为 `8`
- `format` 为 `S24_LE`，即：`PCM_FORMAT_S24_LE`

- `PCM_DEVICE` 的值可以通过 `card0` 中挂在的序号查看，如此处的 `pcm0c` 可知，为 `0`

- 从声卡读取数据实现

`mic_array.c` 的参考代码中给出了音频流数据读取接口的实现，在集成中，根据上述获取的声卡信息，需要将声卡的名称替换成查询到的声卡名称，如下：

```
static int mic_array_device_start_stream(struct mic_array_device_t* dev)
{
    //此处需要替换为对应的声卡名称
    card = find_snd("msm8974-taiko-m");

    if (card < 0) {
        ALOGE("Can't find qualcom iis sound card");
        return -1;
    } else {
        ALOGI("find qualcom iis card with %d", card);
    }

    ...
}
```

如上所示，`find_snd("msm8974-taiko-m")` 需要传入前面查询到的声卡名称，**否则将会无法打开声卡**

1.5 mic_array 模块编译参考

mic_array模块的编译可参考如下代码在系统的源码中进行编译：

```
LOCAL_PATH := $(call my-dir)
include $(CLEAR_VARS)
LOCAL_SRC_FILES := mic_array.c
LOCAL_MODULE_PATH := $(TARGET_OUT_SHARED_LIBRARIES)/hw
LOCAL_MODULE := mic_array.$(TARGET_DEVICE)
LOCAL_MODULE_TAGS := optional
LOCAL_C_INCLUDES += hardware/libhardware \
    external/tinyalsa/include
LOCAL_SHARED_LIBRARIES := liblog libcutils libtinyalsa
LOCAL_MODULE_TARGET_ARCH := arm
include $(BUILD_SHARED_LIBRARY)
```

- 参考如上 `Android.mk` 的编写方式,在Android源码中编译时,会将 `mic_array` 模块编译为 `mic_array.{TARGET_DEVICE}.so` ,生成目录为 `out/target/product/{TARGET_DEVICE}/system/lib/hw/`

二、音频采集程序集成

SDK提供了一个系统级的麦克风阵列音频采集程序，包含如下内容：

- `lothalproc` - 音频采集程序的可执行文件
- `*.so` - 程序所依赖的so库文件

可按如下步骤进行集成：

2.1 集成 `lothalproc`

`lothalproc` 需要作为预编译好的系统的可执行程序集成到 `/system/bin/` 下，可在 `lothalproc` 所在文件夹下的Android.mk中添加如下代码：

```
include $(CLEAR_VARS)
LOCAL_MODULE := lothalproc
LOCAL_MODULE_TAGS := optional
LOCAL_MODULE_CLASS := EXECUTABLES
LOCAL_MODULE_PATH := $(TARGET_OUT)/bin
LOCAL_SRC_FILES := lothalproc
include $(BUILD_PREBUILT)
```

然后在Android的 `/device` 编译配置脚本中（通常对应在 `/device` 目录下对应Vendor和平台下对应的mk文件中）添加如下代码：

```
PRODUCT_PACKAGES += \
    lothalproc
```

2.2 集成so库

在系统编译配置中，需要将这些so库在镜像编译时放入 `/system/lib/` 文件夹下，可在so所在的文件夹下的Android.mk中添加如下代码：

以目录中的 `librlog.so` 为例，需要添加：

```
include $(CLEAR_VARS)
LOCAL_MODULE_TAGS := optional
LOCAL_MODULE := librlog
LOCAL_SRC_FILES := librlog.so
LOCAL_MODULE_SUFFIX := .so
LOCAL_MODULE_CLASS := SHARED_LIBRARIES
include $(BUILD_PREBUILT)
```

同样的，需要添加到系统的`/device`配置中：

```
PRODUCT_PACKAGES += \
    librlog
```

其他的so库按照同样的方式添加，只需要对`LOCAL_MODULE`名和`LOCAL_SRC_FILES`进行修改即可。

2.3 配置init.rc

为了能够在系统启动时加载音频采集程序，同时为了足够的声卡操作权限，需要在系统的init.rc（可能是init.*.rc，根据厂商而定）中添加启动配置，使之以root权限启动，如下：

```
service lothalproc /system/bin/lothalproc <deviceName>
    class main
    user root
    group root root
```

其中`deviceName`为对应系统平台的{TARGET_DEVICE}值，mic_array.{TARGET_DEVICE}.so也必须与之保持一致.比如 deviceName 为 pebble ,则需要如下方式启动

```
service lothalproc /system/bin/lothalproc pebble
    class main
    user root
    group root root
```

三、快速验证

在集成完后，可通过如下方法进行快速验证：

```
$ adb root
```

```
$ adb remount
$ adb push out/target/product/{TARGET_DEVICE}/system/lib/hw/mic_array.$(TARG
ET_DEVICE).so /system/lib/hw/
$ adb push lothalproc /system/bin/
$ adb push floraproc /system/bin/
$ adb shell
$ lothalproc {$TARGET_DEVICE}
```

可通过`adb logcat | grep lothalproc`查看日志是否正常，正常情况下会持续有日志输出。

四、补充说明

4.1 mic_array数据格式

麦克风阵列每秒读取的数据大小取决于`channels` / `rate` / `format` /,比如`format`为`PCM_FORMAT_S32_LE(4 byte)`,`channels`为 8 ,`rate`为 48000,时,麦克风阵列每秒读取数据大小为`48000*8*4 = 1536000(byte)`,麦克风阵列每一次读取数据的大小为`period_count` * `period_size`,每次读取的数据都是分帧组合,每一帧(`frame_size`)大小取决于`channel`以及`format`,比如`channel`为 8,`format`为`PCM_FORMAT_S32_LE`时,每一帧的大小则为`8*4 = 32(byte)`,每一次读取的数据帧数`frame_cnt`即`get_stream_buff_size`方法获取的数据大小为`period_count` * `period_size` / `frame_size`,此大小建议根据麦克风每秒输入数据总大小进行配置,以`10ms/次`的消费速度消费数据.同时每次读取的数据需要根据`format`和`channels`的配置,按照`channels`的顺序进行排列.比如`format`为`PCM_FORMAT_S32_LE`,`channels`为 8,则每帧数据格式如下

```
|32byte(mic0)|32byte(mic1)|32byte(mic2)|32byte(mic3)|32byte(mic4)|32byte(mic
5)|32byte(mic6)|32byte(mic7)|32byte(mic0)|32byte(mic1)|32byte(mic2)|32byte(m
ic3)|32byte(mic4)|32byte(mic5)|32byte(mic6)|32byte(mic7)|.|.|.|.|.|.|.|.|
```

4.2 注意事项

```
static struct hw_module_methods_t mic_array_module_methods = {
    .open = mic_array_device_open,
};

struct mic_array_module_t HAL_MODULE_INFO_SYM = {
    .common = {
        .tag = HARDWARE_MODULE_TAG,
        .version_major = 1,
        .version_minor = 0,
```

```
        .id = "mic_array",
        .name = "mic_array",
        .author = "xxxxxxxx",
        .methods = &mic_array_module_methods,
    },
};
```

- 定义android标准的`HAL_MODULE_INFO_SYM`,其中 `id` 和 `name` 必须使用 `mic_array` ,否则 `lothalproc` 进程在运行时会提示`[Error] Mic Array init (not found mic_array.xxx.so) : -2`