

语音遥控器安卓 6.0+移植指南

应用指南

- 版本号: A_Draft
- 版本日期: 2019-05-14
- 文件编号: MS-PM62-GEN-03-A

修订记录

版本	版本日期	修订人员	修订描述
A_Draft	2019-05-14	李通越 (George)	首次发行;

目录

1 说明	3
1.1 目的	3
1.2 范围	3
1.3 定义	3
2 系统架构	4
3 移植需求	5
4 音频 HAL 移植	7
5 Audio Policy 移植	8
5.1 Service 移植	8
5.2 Configuration 文件修改	12
6 开机启动项修改	14
7 蓝牙连接	15

1 说明

1.1 目的

本文档适用于语音遥控器在安卓 6.0 系统上进行语音移植指导。

1.2 范围

本文档适用于语音遥控器针对安卓 6.0 平台。

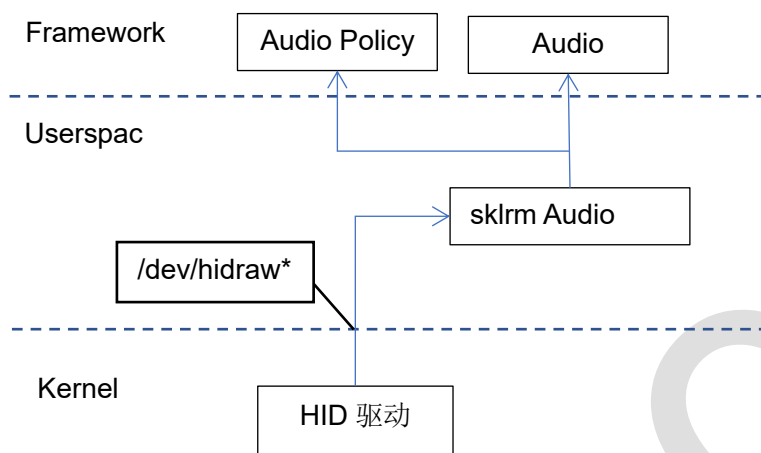
1.3 定义

简称	全称	定义
HAL	Hardware Abstract Layer	硬件抽象层

表 1.1：本文件使用的定义列表

2 系统架构

语音识别助手所调用的音频接口为是安卓通用的音频通道。我司的遥控器语音系统架构为:



当使用语音助手的时候，**audioflinger** 中的 **audiorecorder** 调取相对应遥控器的音频 HAL 库，语音数据通过音频 HAL 接口直接从 **HIDRAW** 节点读取语音数据。

另外语音遥控器也可以支持录音机、微信等使用标准安卓语音通道输入的 **app**。

3 移植需求

在移植之前需要机顶盒端相关平台的系统源码和编译环境支持，并获取 root 权限。现在的移植的安卓平台为安卓 6.0。语音遥控器支持 mstar, amlogic, MTK 等大部分平台。

根据语音通道的系统架构需要对以下部分进行移植：

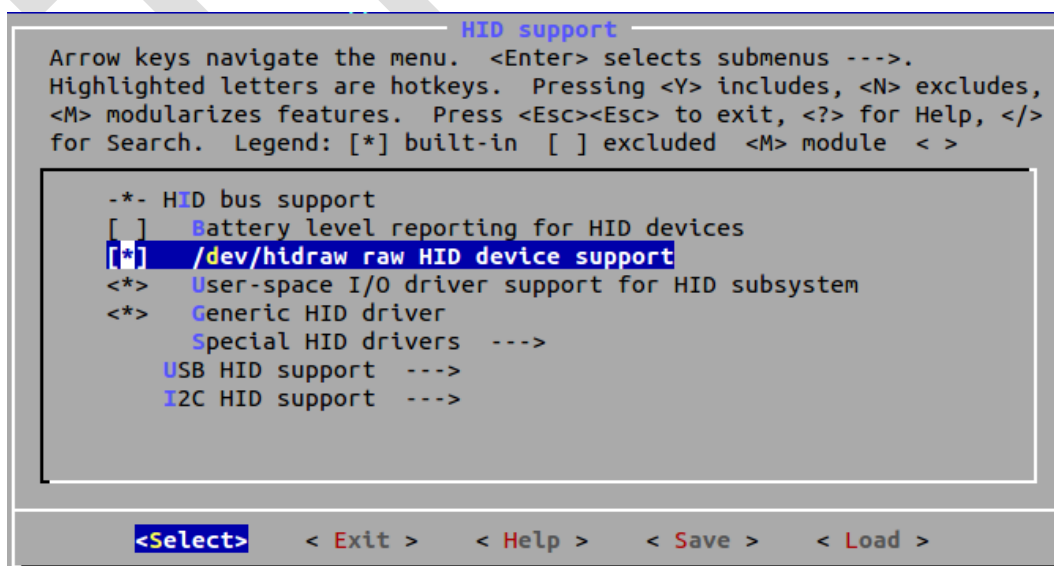
- Audio HAL
- Audio Policy

Audio HAL 位于安卓的 Userspace 用户空间层，直接放入已经编译完成的文件即可；Audio Policy 位于 Framework 固件层，需要修改源码然后重新编译，然后把相应的库刷入机顶盒。

首先打开终端，连接上设备，可以使用 adb 调试指令，adb root 进入 root 模式，使用 adb remount 进行重新挂载；也可以使用串口直接在设备下进行调试，使用 su 进入 root 模式，使用 mount -o remount,rw /system 进行重新挂载。

该语音方案基于 HID profile，需要在移植前确认我司遥控器在安卓设备读取到的 Product ID 和 Vendor ID，并告知我司，以便与我司对软件做相应的修改。

此外，首先要检查当前机顶盒 HIDRAW 功能是否可以正常操作，尤其是在使用 amlogic 平台时候，该功能经常会被关闭。在 HIDRAW 生效时，将遥控器连上以后，安卓设备的/dev 目录下会生成新的 hidraw 节点。如果没有相对应的节点生成，请先检查当前使用内核的 menuconfig 是否配置 hidraw 功能生效。Hidraw 配置在 Device Driver->HID support 下，确认[]raw HID device support 前打上星号。如下图所示：



另外也可以在当前内核的 `config` 文件添加，路径为“`arch/arm/内核版本`”：

```
CONFIG_HIDRAW=y
```

修改后重新编译内核刷入机顶盒重启后即可生效。

4 音频 HAL 移植

语音通道需要音频 hal 来实现具体语音调用方法。在新的音频 HAL 生效时，系统的语音输入通道源会变成遥控器，此时语音数据即可传入安卓的通用语音通道。

此语音库默认的 Vendor ID: 0004 和 Product ID: 0000，如果厂商提供的 PID 和 VID 不相同的话，需要我司重新编译语音库。

在终端输入指令：

```
adb push 路径/audio.skirm.default.so /system/lib/hw
```

把遥控器对应的语音 hal 层库 audio.skirm.default.so:



放在/system/lib/hw 下。

例如：

```
george@george-desktop:~/android/android4.4.2$ adb push out/target/product/rk3288/system/lib/hw/audio.skirm.default.so /system/lib/hw
291 KB/s (13692 bytes in 0.045s)
george@george-desktop:~/android/android4.4.2$
```

然后在该目录下修改文件的权限属性，输入指令：

```
chmod 666 路径/audio.skirm.default.so
```

完成后重启盒子。

如果需要查看该库的 log 信息，可以使用指令：

```
logcat -s audio_hw_skirm
```

5 Audio Policy 移植

在完成了虚拟声卡驱动和音频 hal 的移植，如果需要让新建的 sklrn hal 库生效，还需对 audio policy 进行修改。

5.1 Service 移植

首先在源码下/system/media/audio/include/system/audio.h 中添加新的输入设备：

AUDIO_DEVICE_IN_NEW_MIC 信息，并给该输入设备分配新的设备号。

在 audio_devices_t 结构体中添加

```
AUDIO_DEVICE_IN_NEW_MIC = 0XXXXX,
AUDIO_DEVICE_IN_ALL      = (...|AUDIO_DEVICE_IN_NEW_MIC ),

/* S/PDIF in */
AUDIO_DEVICE_IN_SPDIF      = AUDIO_DEVICE_BIT_IN | 0x10000,
AUDIO_DEVICE_IN_BLUETOOTH_A2DP = AUDIO_DEVICE_BIT_IN | 0x20000,
AUDIO_DEVICE_IN_LOOPBACK   = AUDIO_DEVICE_BIT_IN | 0x40000,
AUDIO_DEVICE_IN_IP          = AUDIO_DEVICE_BIT_IN | 0x80000,

//edit by skl
AUDIO_DEVICE_IN_NEW_MIC      = AUDIO_DEVICE_BIT_IN | 0x100000,

AUDIO_DEVICE_IN_DEFAULT      = AUDIO_DEVICE_BIT_IN | AUDIO_DEVICE_BIT_DEFAULT,

AUDIO_DEVICE_IN_ALL = (AUDIO_DEVICE_IN_COMMUNICATION |
    AUDIO_DEVICE_IN_AMBIENT |
    AUDIO_DEVICE_IN_BUILTIN_MIC |
    AUDIO_DEVICE_IN_BLUETOOTH_SCO_HEADSET |
    AUDIO_DEVICE_IN_WIRED_HEADSET |
    AUDIO_DEVICE_IN_HDMI |
    AUDIO_DEVICE_IN_TELEPHONY_RX |
    AUDIO_DEVICE_IN_BACK_MIC |
    AUDIO_DEVICE_IN_REMOTE_SUBMIX |
    AUDIO_DEVICE_IN_ANLG_DOCK_HEADSET |
    AUDIO_DEVICE_IN_DGTL_DOCK_HEADSET |
    AUDIO_DEVICE_IN_USB_ACCESSORY |
    AUDIO_DEVICE_IN_USB_DEVICE |
    AUDIO_DEVICE_IN_FM_TUNER |
    AUDIO_DEVICE_IN_TV_TUNER |
    AUDIO_DEVICE_IN_LINE |
    AUDIO_DEVICE_IN_SPDIF |
    AUDIO_DEVICE_IN_BLUETOOTH_A2DP |
    AUDIO_DEVICE_IN_LOOPBACK |
    AUDIO_DEVICE_IN_IP |

    //edit by skl
    AUDIO_DEVICE_IN_NEW_MIC |

    AUDIO_DEVICE_IN_DEFAULT),
```


安卓 6.0 将 audio policy 的目录进行了重构，整个 service 的默认路径为：

/frameworks/av/services/audiopolicy。

在打开音频通道时，选择音频输入主要调用的是 `getDeviceForInputSource()` 该函数来获取音频设备的输入源，该函数现在位于以下位置：

/frameworks/av/services/audiopolicy/enginedefault/src/Engine.cpp

在此根据不同的输入需求添加获取遥控器连上 property 的代码：

```
char value[PROPERTY_VALUE_MAX];
int prop_rm;
property_get("audio.in.device.sk1rm", value, NULL);
prop_rm = atoi(value);
if(prop_rm)
    device=AUDIO_DEVICE_IN_NEW_MIC;
```

该设备判断可以添加在不同的音频输入源下，如果需要所有的音频输入源生效可以在如

图位置完成添加：

```
default:
    ALOGW("getDeviceForInputSource() invalid input source %d", inputSource);
    break;
}

//add by skl
char value[PROPERTY_VALUE_MAX];
int prop_rm;
property_get("audio.in.device.sk1rm", value, NULL);
prop_rm = atoi(value);
if(prop_rm)
    device=AUDIO_DEVICE_IN_NEW_MIC;

ALOGV("getDeviceForInputSource()input source %d, device %08x", inputSource, device);
return device;
```

在安卓 6.0 中，`sDeviceNameToEnumTable` 的结构体被移到以下位置：

/frameworks/av/services/audiopolicy/common/managerdefinitions/include/ConfigParsingUtils.h

同样如下添加行：

STRING_TO_ENUM(AUDIO_DEVICE_IN_NEW_MIC),

如图所示：

```
STRING_TO_ENUM(AUDIO_DEVICE_IN_AMBIENT),
STRING_TO_ENUM(AUDIO_DEVICE_IN_BUILTIN_MIC),
STRING_TO_ENUM(AUDIO_DEVICE_IN_BLUETOOTH_SCO_HEADSET),
STRING_TO_ENUM(AUDIO_DEVICE_IN_ALL_SCO),
STRING_TO_ENUM(AUDIO_DEVICE_IN_WIRED_HEADSET),
STRING_TO_ENUM(AUDIO_DEVICE_IN_AUX_DIGITAL),
STRING_TO_ENUM(AUDIO_DEVICE_IN_HDMI),
STRING_TO_ENUM(AUDIO_DEVICE_IN_TELEPHONY_RX),
STRING_TO_ENUM(AUDIO_DEVICE_IN_VOICE_CALL),
STRING_TO_ENUM(AUDIO_DEVICE_IN_BACK_MIC),
STRING_TO_ENUM(AUDIO_DEVICE_IN_REMOTE_SUBMIX),
STRING_TO_ENUM(AUDIO_DEVICE_IN_ANLG_DOCK_HEADSET),
STRING_TO_ENUM(AUDIO_DEVICE_IN_DGTL_DOCK_HEADSET),
STRING_TO_ENUM(AUDIO_DEVICE_IN_USB_ACCESSORY),
STRING_TO_ENUM(AUDIO_DEVICE_IN_USB_DEVICE),
STRING_TO_ENUM(AUDIO_DEVICE_IN_FM_TUNER),
STRING_TO_ENUM(AUDIO_DEVICE_IN_TV_TUNER),
STRING_TO_ENUM(AUDIO_DEVICE_IN_LINE),
STRING_TO_ENUM(AUDIO_DEVICE_IN_SPDIF),
STRING_TO_ENUM(AUDIO_DEVICE_IN_BLUETOOTH_A2DP),
STRING_TO_ENUM(AUDIO_DEVICE_IN_LOOPBACK),
STRING_TO_ENUM(AUDIO_DEVICE_IN_IP),

//edit by skl
STRING_TO_ENUM(AUDIO_DEVICE_IN_NEW_MIC),
```

*另外，在进行 amlogic 平台移植时，会发生和 audio usb 冲突而无法正常初始化音频输入的问题，需要对 amlogic 的部分代码进行修改，才可以正常使用。修改的文件名为 AudioPolicyManager.cpp，路径为：

```
/frameworks/av/services/audiopolicy/managerdefault/AudioPolicyManager.cpp
```

找到 getInputForAttr(...)和 setInputDevice(...)这两个与输入设备相关的函数，在 amlogic 添加的部分中，将判断 device 是否为 AUDIO_DEVICE_IN_USB_DEVICE 的方式从 and 改为 equal：


```
if (device & AUDIO_DEVICE_IN_USB_DEVICE) => if (device ==
    AUDIO_DEVICE_IN_USB_DEVICE)
```


如下图进行修改即可：

<pre>//Amlogic if (device == AUDIO_DEVICE_IN_USB_DEVICE) { DeviceVector deviceList; deviceList = mAvailableInputDevices.getDevicesFromType(device); ALOG_ASSERT(!deviceList.isEmpty(), "getInputForAttr no device"); char addr[PROPERTY_VALUE_MAX]; property_get("audio.usb.input.setting", addr, "null"); size_t i = 0; for (i = 0; i < deviceList.size(); i++) { ALOGV("deviceList i = %d, addr = %s ", i, deviceList.itemAt(i)->mAddress.string()); if (strncmp(deviceList.itemAt(i)->mAddress.string(), addr, strlen(addr)) == 0) { ALOGI("audio.usb.input.setting: %s, checked OK", addr); break; } } if (i >= deviceList.size()) i = 0; address = deviceList.itemAt(i)->mAddress; } //endof Amlogic //Amlogic size_t i = 0; if (device == AUDIO_DEVICE_IN_USB_DEVICE) { char address[PROPERTY_VALUE_MAX]; property_get("audio.usb.input.setting", address, "null"); size_t i = 0; for (i = 0; i < deviceList.size(); i++) { ALOGV("deviceList i = %d, addr = %s ", i, deviceList.itemAt(i)->mAddress.string()); if (strncmp(deviceList.itemAt(i)->mAddress.string(), address, strlen(address)) == 0) { ALOGI("audio.usb.input.setting: %s, checked OK", address); break; } } if (i >= deviceList.size()) i = 0; } //endof Amlogic</pre>	<pre>//Amlogic if (device & AUDIO_DEVICE_IN_USB_DEVICE) { DeviceVector deviceList; deviceList = mAvailableInputDevices.getDevicesFromType(device); ALOG_ASSERT(!deviceList.isEmpty(), "getInputForAttr no device"); char addr[PROPERTY_VALUE_MAX]; property_get("audio.usb.input.setting", addr, "null"); size_t i = 0; for (i = 0; i < deviceList.size(); i++) { ALOGV("deviceList i = %d, addr = %s ", i, deviceList.itemAt(i)->mAddress.string()); if (strncmp(deviceList.itemAt(i)->mAddress.string(), addr, strlen(addr)) == 0) { ALOGI("audio.usb.input.setting: %s, checked OK", addr); break; } } if (i >= deviceList.size()) i = 0; address = deviceList.itemAt(i)->mAddress; } //endof Amlogic //Amlogic size_t i = 0; if (device & AUDIO_DEVICE_IN_USB_DEVICE) { char address[PROPERTY_VALUE_MAX]; property_get("audio.usb.input.setting", address, "null"); size_t i = 0; for (i = 0; i < deviceList.size(); i++) { ALOGV("deviceList i = %d, addr = %s ", i, deviceList.itemAt(i)->mAddress.string()); if (strncmp(deviceList.itemAt(i)->mAddress.string(), address, strlen(address)) == 0) { ALOGI("audio.usb.input.setting: %s, checked OK", address); break; } } if (i >= deviceList.size()) i = 0; } //endof Amlogic</pre>
---	---


按照此图进行修改，左边为修改后的代码，右边为修改前的代码，在修改完成后将会避免该问题的发生。

然后重新编译整个 **audiopolicy** 的目录，根据系统需要使用 32 位或者 64 位的编译方式，生成以下新的库：


libaudiopolicyengine
nedefault.so


libaudiopolicymanager.
so


libaudiopolicymanager
default.so


libaudiopolicyservice.
so

将以下新编译生成的库放在 **/system/lib** 下，如果是 64 位的库则放在 **/system/lib64** 下，在终端输入指令：

```
adb push 路径/libaudiopolicyengine/default.so /system/lib
```

```
adb push 路径/libaudiopolicymanager.so /system/lib
```

```
adb push 路径/libaudiopolicymanager/default.so /system/lib
```

```
adb push 路径/libaudiopolicyservice.so /system/lib
```

即可。

5.2 Configuration 文件修改

另外 audio policy 还需要修改系统下的 audio_policy.conf:



该文件一般位于/device/厂商/路径下，把新建的 sklrn 模块的信息添加进去然后保存。

```
sklrn {
    inputs {
        sklrn {
            sampling_rates 16000
            channel_masks AUDIO_CHANNEL_IN_MONO
            formats AUDIO_FORMAT_PCM_16_BIT
            devices AUDIO_DEVICE_IN_NEW_MIC
        }
    }
}
```

如图添加完:

```
audio_hw_modules {
    primary {
        outputs {
            primary {
                sampling_rates 48000
                channel_masks AUDIO_CHANNEL_OUT_STEREO
                formats AUDIO_FORMAT_PCM_16_BIT
                devices AUDIO_DEVICE_OUT_SPEAKER|AUDIO_DEVICE_OUT_WIRED_HEADSET
                flags AUDIO_OUTPUT_FLAG_PRIMARY
            }
        }
        inputs {
            primary {
                sampling_rates 8000|11025|16000|22050|24000|32000|44100|48000
                channel_masks AUDIO_CHANNEL_IN_MONO
                formats AUDIO_FORMAT_PCM_16_BIT
                devices AUDIO_DEVICE_IN_BUILTIN_MIC|AUDIO_DEVICE_IN_USB_MIC|AUDIO_DEVICE_IN_WIRED_HEADSET
            }
        }
    }
}

sklrn {
    inputs {
        sklrn {
            sampling_rates 16000
            channel_masks AUDIO_CHANNEL_IN_MONO
            formats AUDIO_FORMAT_PCM_16_BIT
            devices AUDIO_DEVICE_IN_NEW_MIC
        }
    }
}
```

然后把更新后的 `audio_policy.conf` 放在 `/system/etc` 下，在终端输入指令：

```
adb push 路径/audio_policy.conf /system/etc
```

或者重新编译镜像代码并刷入。

Tropos

6 开机启动项修改

在调用语音库的时候，会对 **hidraw** 节点进行操作，需要先修改该节点的权限，否则会发生权限被拒绝的问题，具体操作为对系统文件 **ueventd.rc** 进行修改。

在 **ueventd.rc** 下添加：

```
/dev/hidraw* 0666 root root
```

另外要在开机时候自动让语音库生效需要设置属性，在开机启动项 **init.rc** 中添加 **property** 属性：

```
setprop audio.in.device.skIrm 1
```

或者修改启动的 **.prop** 文件，添加：

```
audio.in.device.skIrm = 1
```

在完成对该两文件的修改后，重新编译生成 **boot.img** 或者整个镜像文件刷入，完成以上所有步骤后，重启机顶盒，语音识别移植即可成功。

7 蓝牙连接

一般情况下，如果需要正常的语音使用，遥控器端与平台端蓝牙的连接参数需要互相匹配。平台端的连接参数文件为 `btm_ble_api.h`。

安卓 6.0 的路径为： `/system/bt/stack/include/btm_ble_api.h`

需要修改最小和最大连接 `interval`，从机的 `latency` 以及 `timeout`。

其中我司建议修改以下数值：

```
#define BTM_BLE_CONN_INT_MIN 0x0006
#define BTM_BLE_CONN_INT_MIN_DEF 6
#define BTM_BLE_CONN_INT_MAX_DEF 6
#define BTM_BLE_CONN_SLAVE_LATENCY_DEF 10
#define BTM_BLE_CONN_TIMEOUT_DEF 100
```

修改后如图所示：

```
/* default connection interval min */
#ifndef BTM_BLE_CONN_INT_MIN_DEF
#define BTM_BLE_CONN_INT_MIN_DEF 6 /* recommended min: 7.5ms = 6 * 1.25 */
#endif
/* default connectino interval max */
#ifndef BTM_BLE_CONN_INT_MAX_DEF
#define BTM_BLE_CONN_INT_MAX_DEF 6 /* recommended max: 7.5ms = 6 * 1.25 */
#endif
/* default slave latency */
#ifndef BTM_BLE_CONN_SLAVE_LATENCY_DEF
#define BTM_BLE_CONN_SLAVE_LATENCY_DEF 10 /* 10 */
#endif
/* default supervision timeout */
#ifndef BTM_BLE_CONN_TIMEOUT_DEF
#define BTM_BLE_CONN_TIMEOUT_DEF 100
#endif
```

然后重新编译生成新的蓝牙库文件：



bluetooth.default.
so

在终端输入指令：

`adb push 路径/bluetooth.default.so /system/lib/hw`

把新编译生成的蓝牙库 `bluetooth.default.so` 放在 `/system/lib/hw` 下重启后即可。