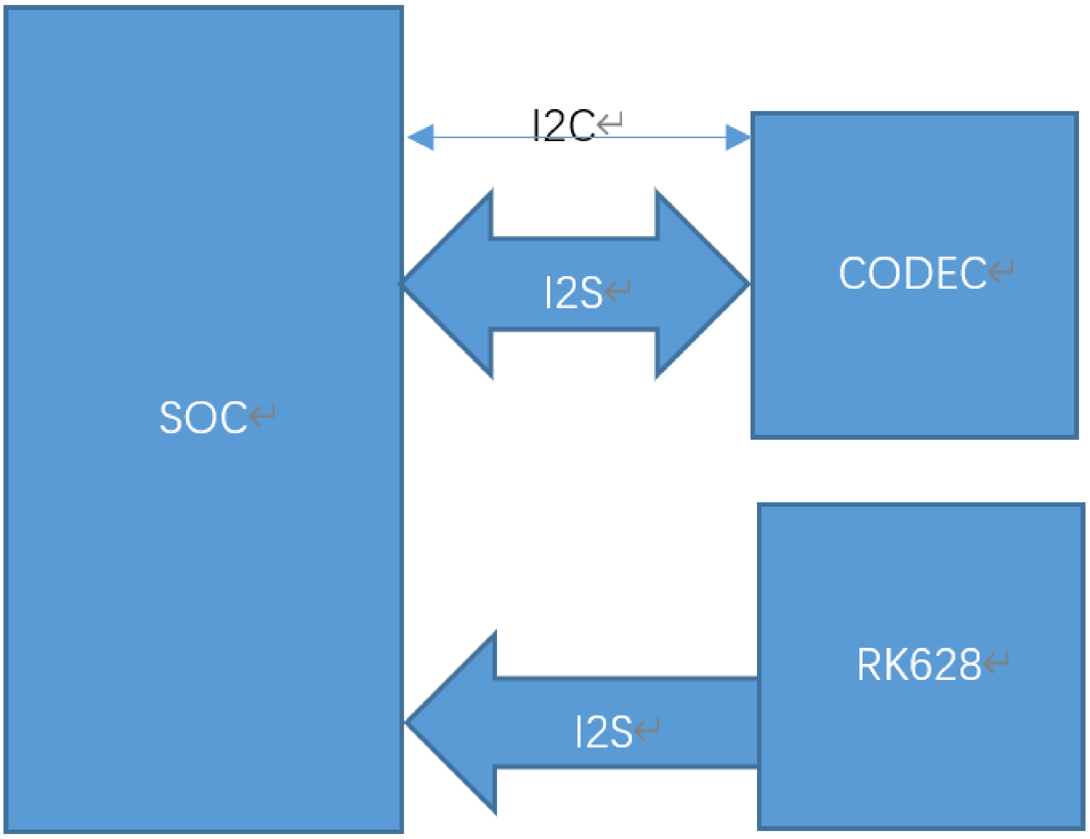
RK628 直接连接SOC的情况



直接连接有几个问题点：

1 音频采样率是跟HDMITX的音频采样率是一致的，而APK采样率是APK自行设置的，所以在HAL层需要一个重采样过程

1. KERNEL

RK628直连SOC的时候，不需要额外CODEC配置，默认情况下，628会直接输出I2S信号，SOC设置好I2S格式就好了,需要配置SOC I2S为从模式：

```
dummy_codec: dummy-codec {
        compatible = "rockchip,dummy-codec";
        #sound-dai-cells = <0>;
};

hdmiin-sound {
        compatible = "simple-audio-card";
        simple-audio-card,format = "i2s";
        simple-audio-card,name = "rockchip,hdmiin";
        simple-audio-card,bitclock-master = <&dailink0_master>;
        simple-audio-card,frame-master = <&dailink0_master>;
        status = "okay";
        simple-audio-card,cpu {
                sound-dai = <&i2s0>;
        };
        dailink0_master: simple-audio-card,codec {
                sound-dai = <&dummy_codec>;
        };
};
```

2. HARDWARE

2.1 当前提交节点

关于hal层部分，建议更新以下hal代码，关注这个提交，这个主要是HAL层根据声卡的信息获取对应声卡设备的提交

早期的解析，需要提前配置好 card以及device，不够灵活，建议更新下：

```
commit 10fb966e1931025f6f369ceff9ac81d7c6aac1f1
Author: Shunhua Lan <lsh@rock-chips.com>
Date:   Mon Dec 2 17:00:21 2019 +0800

    [audio hal]: add device parse when selecting specific sound card

    Change-Id: I76727d006045a10191aae5c427f7cb87b534f18c
    Signed-off-by: Shunhua Lan <lsh@rock-chips.com>
```

关于HAL代码，这边直接压缩了一份： audiohal-202104121040.zip
MD5:007bd5883851b3109f168f3f4ca96dfb

这个提交，主要关注以下信息：

1. HDMI_IN_NAME: HDMI IN设备匹配信息

```
struct dev_proc_info HDMI_IN_NAME[] =
{
    {"realtekrt5651co", "tc358749x-audio"},
    {"rockchiprt5640c", NULL},
    {NULL, NULL}, /* Note! Must end with NULL, else will cause crash */
};
```

这个信息主要是根据声卡的以下信息匹配：

```
rk3288_Android10:/ # cat /proc/asound/cards
 0 [rockchiprt5640c]: rockchiprt5640c - rockchiprt5640codec_hdmiin
                      rockchiprt5640codec_hdmiin
```

如上，用户可以通过cat /proc/asound/cards获取对应的HDMI IN声卡的信息，然后把 [XXXXXX] 里面的信息添加到数组HDMI_IN_NAME第一个成员里面，第二个直接赋值为NULL就可以了。

2.2 补丁文件

同步以上提交之后，需要添加的补丁：

0001-support-HDMIIn-capture-mode.patch MD5: 6763ca0404cc44add81d3802e56d1822

以下对补丁进行解析

2.2.1 0001-support-HDMIIn-capture-mode.patch

这个补丁主要是添加一个**HDMI_IN_CAPTURE_ROUTE**配置，这个配置的作用就是配置从HDMIN IN声卡录取音频数据。当打开应用录音时候，audiopolicy选择打开**AUDIO_DEVICE_IN_HDMI**设备，这个时候，我们选择打开的声卡是**HDMI_IN_NAME**数组里面定义的设备：

```
struct dev_proc_info HDMI_IN_NAME[] =
{
    {"realtekrt5651co", "tc358749x-audio"},
    {NULL, NULL}, /* Note! Must end with NULL, else will cause crash */
};
```

添加了获取**AUDIO_DEVICE_IN_HDMI**输入设备采样率的代码，主要是用于打开声卡时候，配置声卡的采样率，这个采样率由应用通过配置**vendor.hdmiin.audiorate**的属性获取：

```
#define STR_32KHZ "32KHZ"
#define STR_44_1KHZ "44.1KHZ"
#define STR_48KHZ "48KHZ"
/**
 * @brief get_hdmiin_audio_rate
 * @param
 * @return hdmiin audio rate
 */
static int get_hdmiin_audio_rate(struct audio_device *adev)
{
    int rate = 44100;
    char value[PROPERTY_VALUE_MAX] = "";
    property_get("vendor.hdmiin.audiorate", value, STR_44_1KHZ);

    if ( 0 == strncmp(value, STR_32KHZ, strlen(STR_32KHZ)) ){
        rate = 32000;
    }else if( 0 == strncmp(value, STR_44_1KHZ, strlen(STR_44_1KHZ)) ){
        rate = 44100;
    }else if( 0 == strncmp(value, STR_48KHZ, strlen(STR_48KHZ)) ){
        rate = 48000;
    } else {
        rate = atoi(value);
        if (rate <= 0)
            rate = 44100;
    }
    // if hdmiin connect to codec, use 44100 sample rate
    if (adev->dev_in[SND_IN_SOUND_CARD_HDMI].card
            == adev->dev_out[SND_OUT_SOUND_CARD_SPEAKER].card)
        rate = 44100;

    return rate;
}
```

2.3 添加具体支持的声卡信息

hdmiin设备的声卡信息是在audio_hw.c源文件里面，通过**HDMI_IN_NAME**数组进行匹配的，如下：

- cid： 声卡的名字
- did：声卡对应cocec dai的名字，一般只有一个设备的声卡，这里直接设置为NULL就可以了

```
struct dev_proc_info
{
    const char *cid; /* cardX/id match */
    const char *did; /* dai id match */
};

struct dev_proc_info HDMI_IN_NAME[] =
{
    {"realtekrt5651co", "tc358749x-audio"},
    {"rockchiprt5640c", NULL},
    {NULL, NULL}, /* Note! Must end with NULL, else will cause crash */
};
```

这样，在开始录音时候，hal就能根据系统所选设备，打开对应的声卡

3. FRAMEWORK

这里面的操作就是打两个补丁，设置音频输入、输出策略，添加HDMI IN设备，补丁如下:

frameworks_av_support_hdmiin.patch MD5: 4227e831be0dfe00d11d3fd83f57848c

Android11的补丁： 0001-support-hdmiin.patch

device_rockchip_common_support_hdmiin.patch MD5: fd27c82b30c074969f42916e3fc3bf09

补丁主要是修改下面两个文件：

- Engine.cpp：设置音频输入、输出策略
- audio_policy_configuration.xml：添加HDMI IN设备

3.1 Engine.cpp - audiopolicy

主要是修改Engine.cpp文件，决定audiopolicy使用具体的输入输出设备

系统属性 `media.audio.device_policy`，设置当前优先输出设备的类型，可以是以下设备：

```
private static final String[] strOutDevice = {
    "bypass",                        // bypass to SPK in codec
    "hdmi",                          // output to hdmi
    "speaker",                       // output to SPK
    "usb",                           // output to usb audio
    "bluetooth",                     // output to bluetooth
    "hdmi,speaker,usb,bluetooth",    // output to all
    ""                               // auto select audio device
};
```

系统属性 `media.audio.hdmiin`，设置当前输入设备的类型，这里面，对于使用HDMI IN的时候，都会被设置为 `true`。然后audiopolicy打开的录音设备就是AUDIO_DEVICE_IN_HDMI。这个类型会被传入到HAL，HAL再具体操作HDMIN声卡。

实例：对于使用HDMIIN设备录音，HDMIOUT 设备播放，应用可以通过以下方法设置：

```
SystemProperties.set("media.audio.device_policy", "hdmi");
SystemProperties.set("media.audio.hdmiin", "true");
```

frameworks_av_support_hdmiin.patch的补丁内容：

```
diff --git a/services/audiopolicy/enginedefault/src/Engine.cpp
b/services/audiopolicy/enginedefault/src/Engine.cpp
old mode 100644
new mode 100755
index 3e13e50..24df9a4
--- a/services/audiopolicy/enginedefault/src/Engine.cpp
+++ b/services/audiopolicy/enginedefault/src/Engine.cpp
@@ -31,6 +31,7 @@
 #include <policy.h>
 #include <utils/String8.h>
 #include <utils/Log.h>
+#include <cutils/properties.h>

 namespace android
 {
@@ -606,6 +607,55 @@ audio_devices_t
Engine::getDeviceForStrategyInt(routing_strategy strategy,
         ALOGE_IF(device == AUDIO_DEVICE_NONE,
                  "getDeviceForStrategy() no default device defined");
     }
+
+    char value[PROPERTY_VALUE_MAX];
+    property_get("media.audio.device_policy", value, "");
+    if (value[0]) {
+        uint32_t new_device = AUDIO_DEVICE_NONE;
+        if (strstr(value, "hdmi")) {
+            ALOGD("set audio policy to hdmi, availableOutputDevicesType :
0x%x", availableOutputDevicesType);
+            if (availableOutputDevicesType & AUDIO_DEVICE_OUT_HDMI) {
+                ALOGD("set audio policy to hdmi succeed");
+                new_device |= AUDIO_DEVICE_OUT_HDMI;
+            }
+        }
+        if (strstr(value, "spdif")) {
+            ALOGD("set audio policy to spdif, availableOutputDevicesType :
0x%x", availableOutputDevicesType);
+            new_device |= AUDIO_DEVICE_OUT_SPDIF;
+        }
+        if (strstr(value, "speaker")) {
+            ALOGD("set audio policy to speaker, availableOutputDevicesType :
0x%x", availableOutputDevicesType);
+            if (availableOutputDevicesType & AUDIO_DEVICE_OUT_SPEAKER) {
+                ALOGD("set audio policy to speaker succeed");
+                new_device |= AUDIO_DEVICE_OUT_SPEAKER;
+            }
+        }
+        if (strstr(value, "usb")) {
+            ALOGD("set audio policy to usb, availableOutputDevicesType : 0x%x",
availableOutputDevicesType);
+            if (availableOutputDevicesType & AUDIO_DEVICE_OUT_USB_DEVICE) {
+                ALOGD("set audio policy to usb succeed");
+                new_device |= AUDIO_DEVICE_OUT_USB_DEVICE;
+            }
+            else if (availableOutputDevicesType & AUDIO_DEVICE_OUT_USB_HEADSET)
{
+                ALOGD("set audio policy to usb succeed");
+                new_device |= AUDIO_DEVICE_OUT_USB_HEADSET;
+            }
+        }
+        if (strstr(value, "bluetooth")) {
+            ALOGD("set audio policy to bluetooth, availableOutputDevicesType :
0x%x", availableOutputDevicesType);
+            if (availableOutputDevicesType & AUDIO_DEVICE_OUT_BLUETOOTH_A2DP)
+                new_device |= AUDIO_DEVICE_OUT_BLUETOOTH_A2DP;
+            else if (availableOutputDevicesType &
AUDIO_DEVICE_OUT_BLUETOOTH_A2DP_HEADPHONES)
+                new_device |= AUDIO_DEVICE_OUT_BLUETOOTH_A2DP_HEADPHONES;
+            else if (availableOutputDevicesType &
AUDIO_DEVICE_OUT_BLUETOOTH_A2DP_SPEAKER)
+                new_device |= AUDIO_DEVICE_OUT_BLUETOOTH_A2DP_SPEAKER;
+        }
+
+        if (new_device != AUDIO_DEVICE_NONE) {
+            device = new_device;
+        }
+    }
+
     ALOGVV("getDeviceForStrategy() strategy %d, device %x", strategy, device);
     return device;
 }
@@ -722,6 +772,10 @@ audio_devices_t
Engine::getDeviceForInputSource(audio_source_t inputSource) cons
         }
         break;
     case AUDIO_SOURCE_CAMCORDER:
+        if ((availableDeviceTypes & AUDIO_DEVICE_IN_HDMI)
+                && property_get_bool("media.audio.hdmiin", false)) {
+            device = AUDIO_DEVICE_IN_HDMI;
+        }else
         if (availableDeviceTypes & AUDIO_DEVICE_IN_BACK_MIC) {
             device = AUDIO_DEVICE_IN_BACK_MIC;
         } else if (availableDeviceTypes & AUDIO_DEVICE_IN_BUILTIN_MIC) {
```

3.2 audio_policy_configuration.xml

device_rockchip_common_support_hdmiin.patch的补丁内容

```
diff --git a/audio_policy_configuration.xml b/audio_policy_configuration.xml
index a49cd9a..1202c75 100644
--- a/audio_policy_configuration.xml
+++ b/audio_policy_configuration.xml
@@ -22,6 +22,7 @@
             <attachedDevices>
                 <item>Speaker</item>
                 <item>Built-In Mic</item>
+                <item>HDMIIn</item>
             </attachedDevices>
             <defaultOutputDevice>Speaker</defaultOutputDevice>
             <mixPorts>
@@ -69,6 +70,8 @@
                 </devicePort>
                 <devicePort tagName="BT SCO Headset Mic"
type="AUDIO_DEVICE_IN_BLUETOOTH_SCO_HEADSET" role="source">
                 </devicePort>
+                <devicePort tagName="HDMIIn" type="AUDIO_DEVICE_IN_HDMI"
role="source">
+                </devicePort>
             </devicePorts>
             <routes>
                 <route type="mix" sink="Speaker"
@@ -89,7 +92,7 @@
                        sources="primary output,spdif_passthrough"/>

                 <route type="mix" sink="primary input"
-                       sources="Built-In Mic,Wired Headset Mic,BT SCO Headset
Mic"/>
+                       sources="Built-In Mic,Wired Headset Mic,BT SCO Headset
Mic,HDMIIn"/>
             </routes>
         </module>
```

4. 实际应用

5. 总结

对于不同的应用场景，有几个是必须的操作：

1. 内核声卡的补丁

   这个如果不会配置，可以联系RK音频工程师

2. HARDWARE 的补丁

   0001-support-HDMIIn-capture-mode.patch

3. FRAMEWORK的两个补丁：

   frameworks_av_support_hdmiin.patch

   device_rockchip_common_support_hdmiin.patch

4. 应用程序