[Linux USB 3.0驱动分析（七）——UAC驱动分析](https://www.cnblogs.com/wen123456/p/14281917.html)

因为项目里面有USB音频外设，所以需要分析一下UAC驱动。

USB Audio Class，USB音频类，一个像USB这样的通用数据接口，可以有很多种实现数字音频数据传输的方式。不同的开发者可以根据自己的喜好和需求，定义任意的控制方式，传输模式，音频格式等等参数。

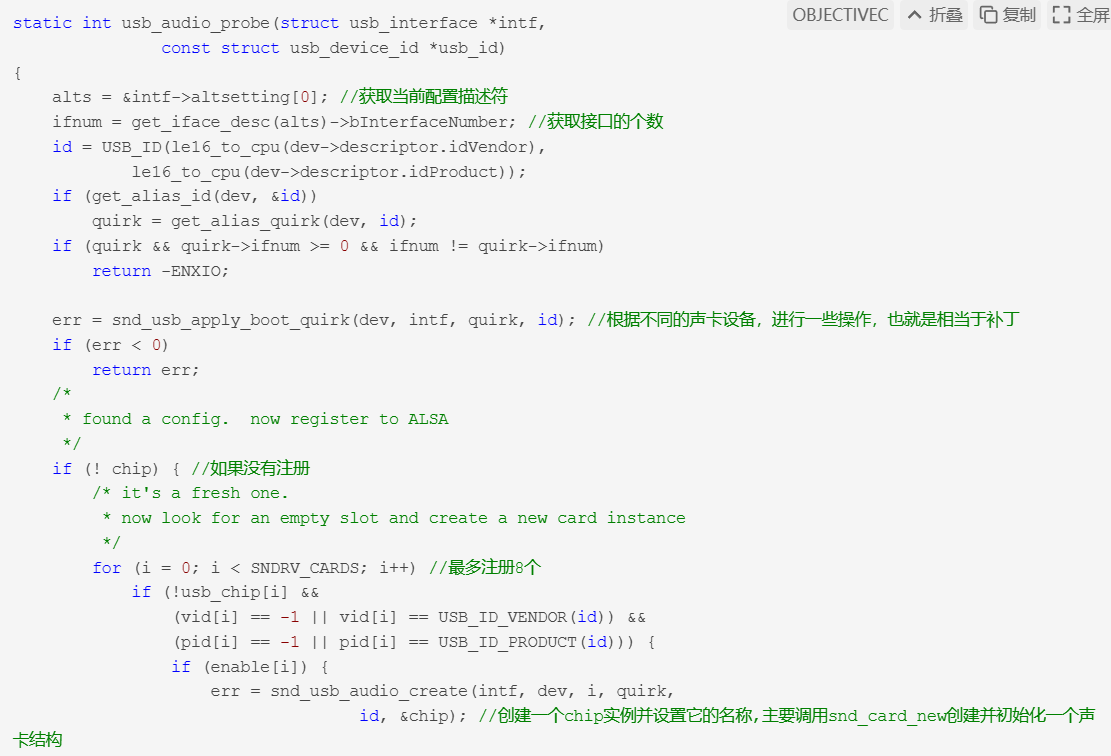
**一.UAC驱动初始化分析**

**代码路径：sound\usb\card.c**

我们先来看看初始化部分，主要是初始化usb\_audio\_driver结构体

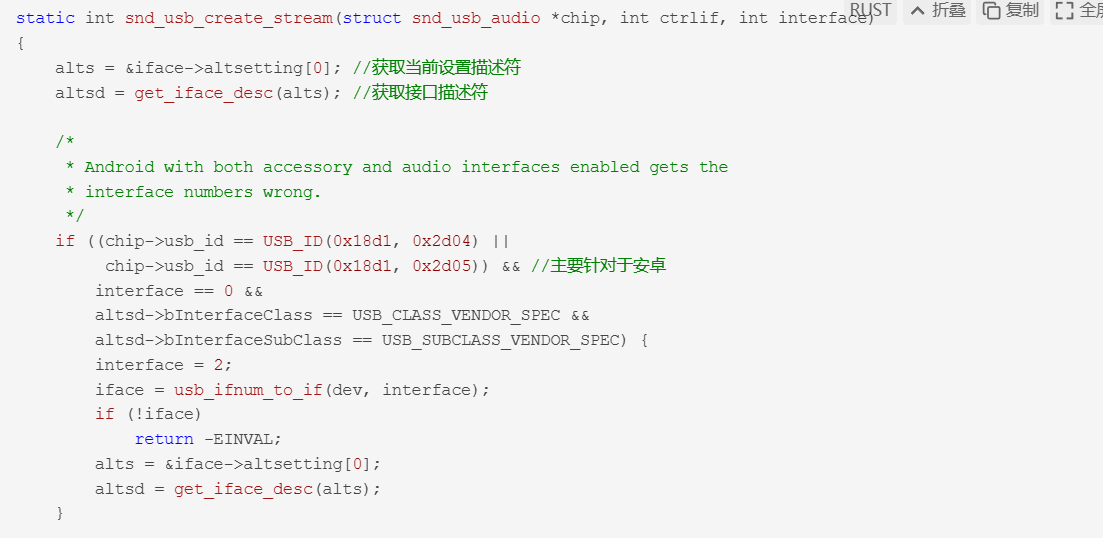


匹配过后，进入usb\_audio\_probe函数里面





我们重点分析一下snd\_usb\_create\_streams,它主要调用snd\_usb\_create\_stream，主要是根据不同的协议，进入不同的分支



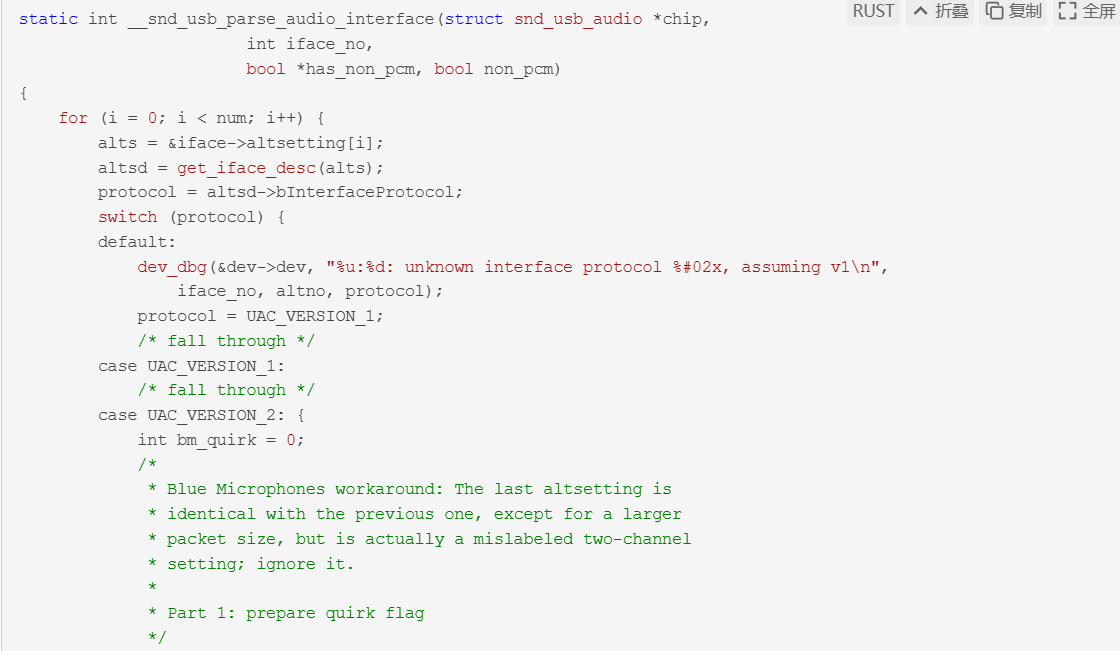


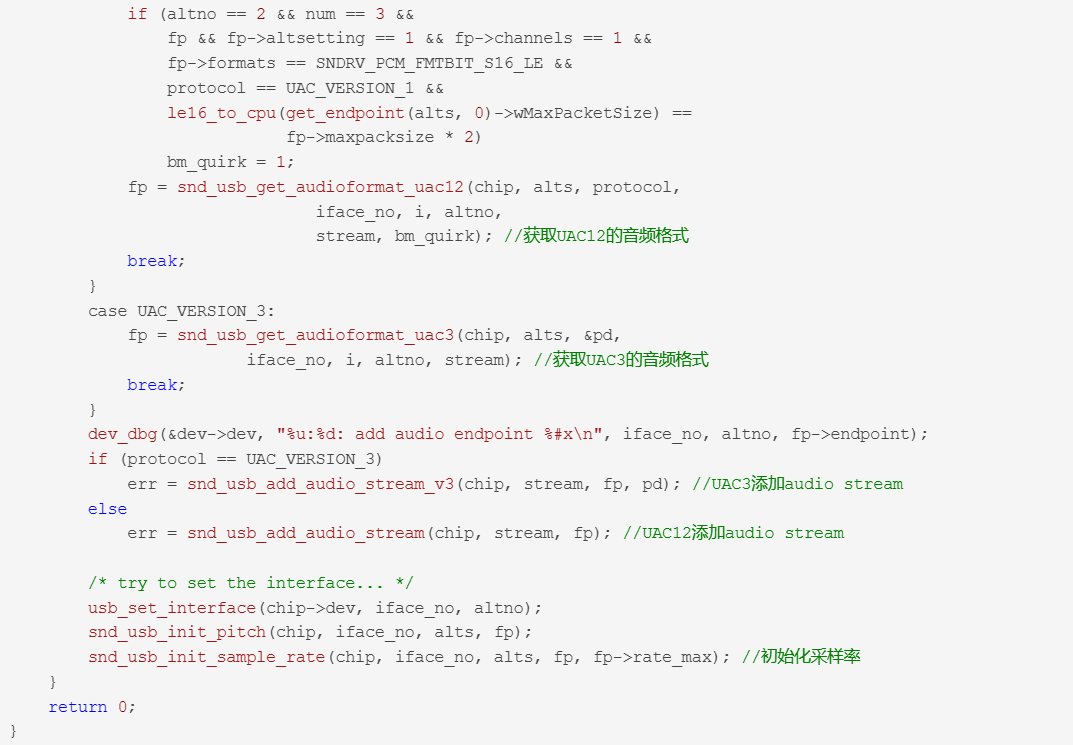
这里面有一个重要的函数，snd\_usb\_parse\_audio\_interface，用来解析音频接口，也就是pcm。这里面UAC1.0,2.0和UAC3.0有些一样。

UAC1.0,2.0使用snd\_usb\_add\_audio\_stream，UAC3.0使用snd\_usb\_add\_audio\_stream\_v3。我们只看UAC1.0,2.0

大概的流程是

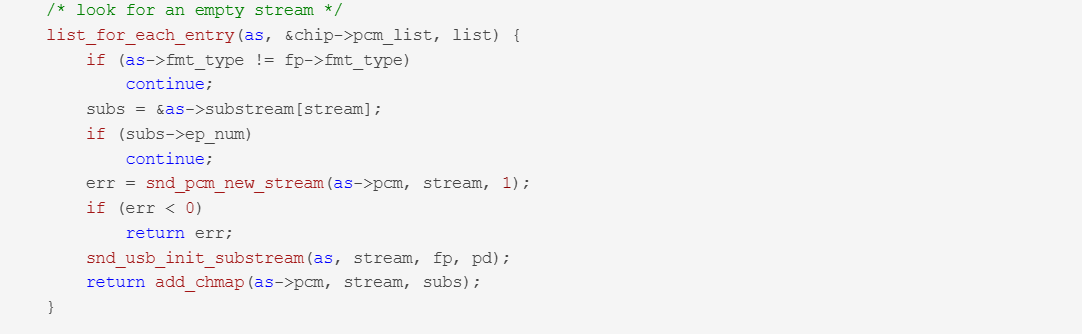
**snd\_usb\_parse\_audio\_interface -> snd\_usb\_parse\_audio\_interface**

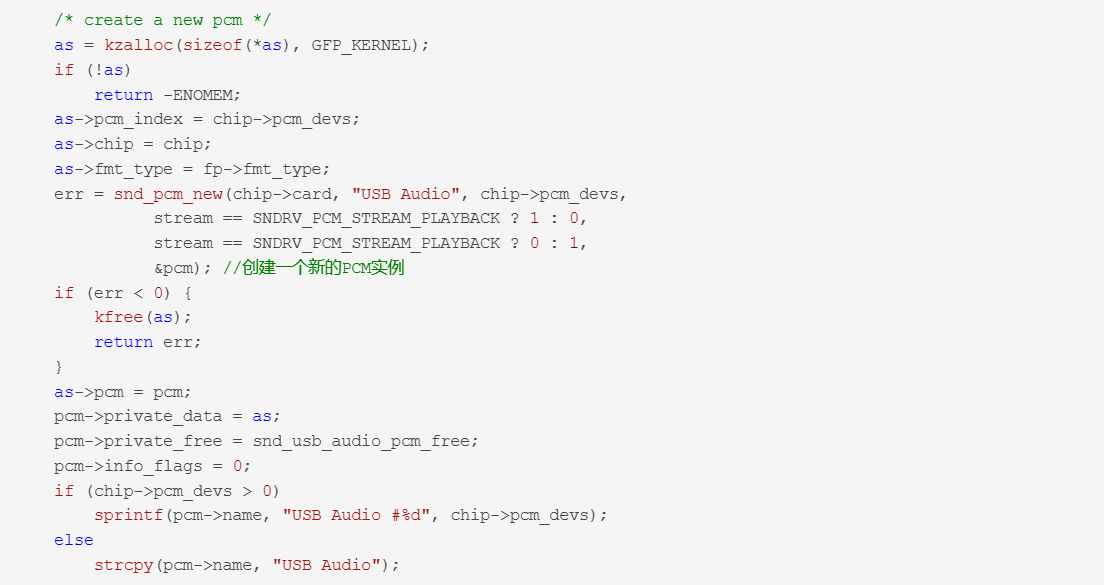


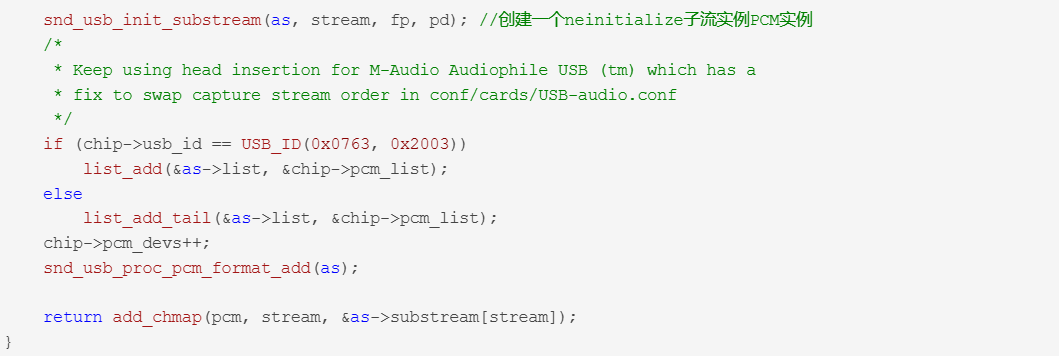


snd\_usb\_add\_audio\_stream和snd\_usb\_add\_audio\_stream\_v3都会调用\_\_snd\_usb\_add\_audio\_stream，这里创建stream.该接口函数中根据stream的类型会分别创建playback\_stream和capture\_stream，并分别创建对应的playback的pcm接口和capture的接口：

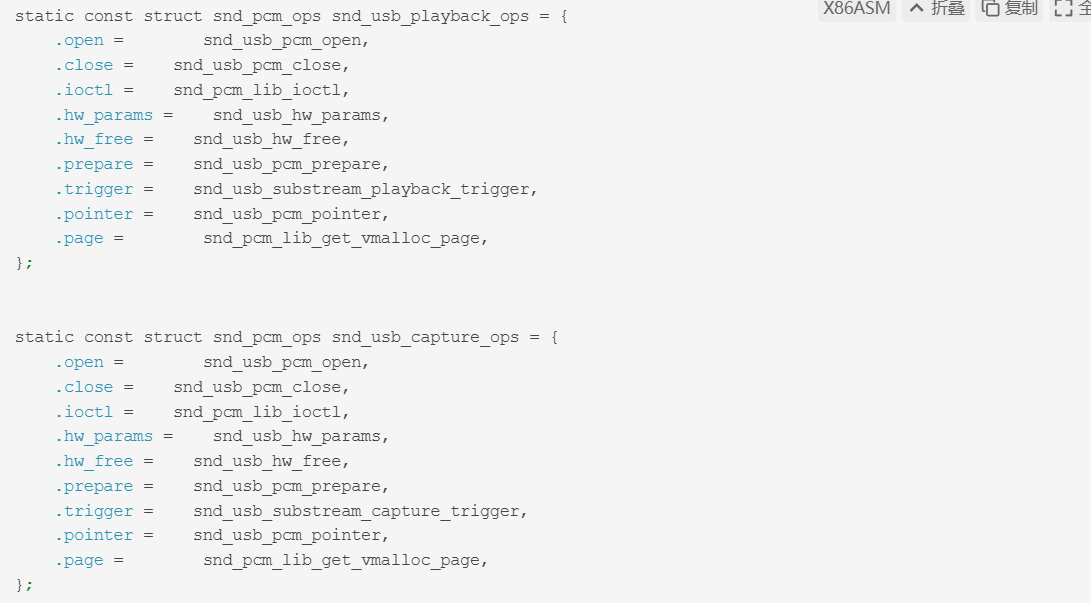


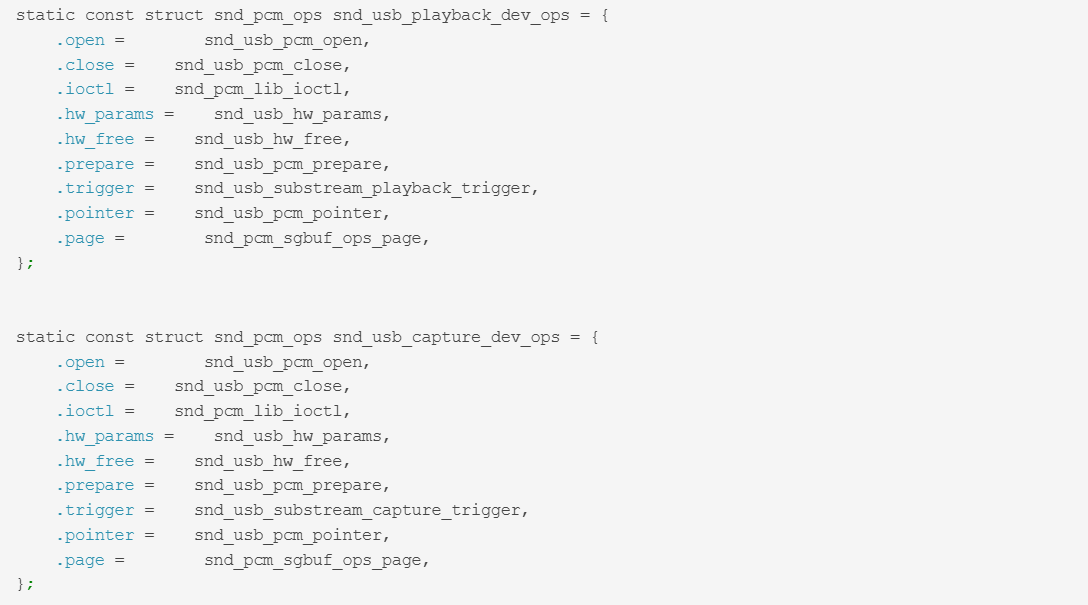


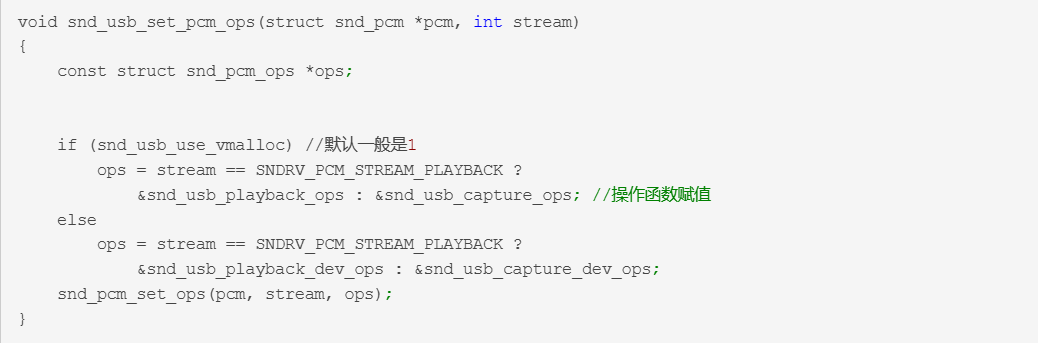




其中最重要的是snd\_usb\_set\_pcm\_ops()函数，它设置了pcm\_substream的操作接口函数.







snd\_usb\_use\_vmalloc默认是1，上面函数给pcm->substream->ops赋值，根据stream类型，分别赋值为snd\_usb\_playback\_ops 或者 snd\_usb\_capture\_ops。到这里初始化基本上已经完成了。

**二.UAC驱动读写分析**

**ALSA音频播放之前已经有分析过来，具体参考《  rk音频驱动分析之tinyplay播放》**

前面部分都是一样的，只是调用到不同的substream调用不同的substream的ops函数

**1.open**

应用打开设备：open(fn, O\_RDWR);  //比如是/dev/snd/pcmC0D0p

对于UAC设备通过一系列掉用就会到snd\_usb\_pcm\_open

主要是一些结构体初始化，还有建立硬件方面的信息。snd\_usb\_stream下面有snd\_usb\_substream，snd\_usb\_substream下面有snd\_pcm\_substream



**2.设置音频参数**

应用调用ioctl：ioctl(pcm->fd, SNDRV\_PCM\_IOCTL\_HW\_PARAMS, &params) //设置音频硬件参数

对于UAC设备通过一系列掉用就会到snd\_usb\_hw\_params，主要是设置一些音频参数，还有接口和端点方面的设置。



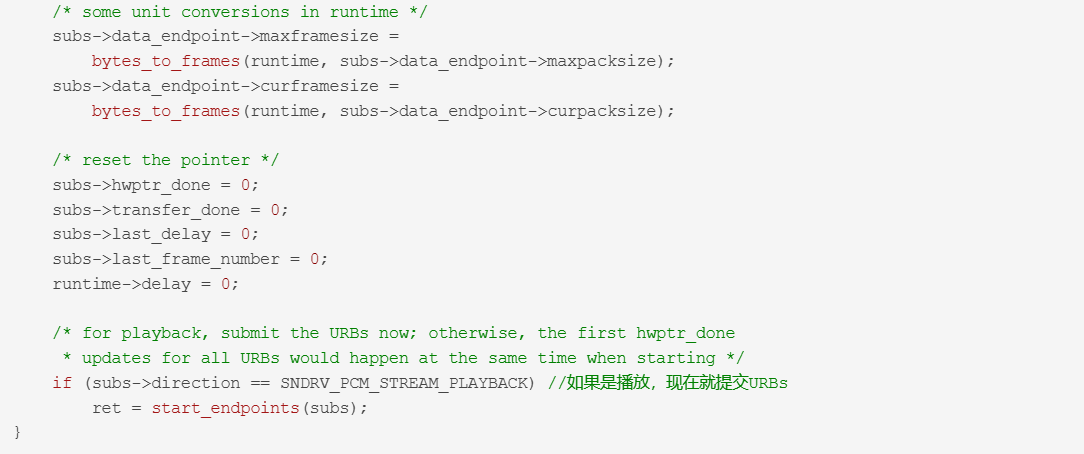


**3.预备传输**

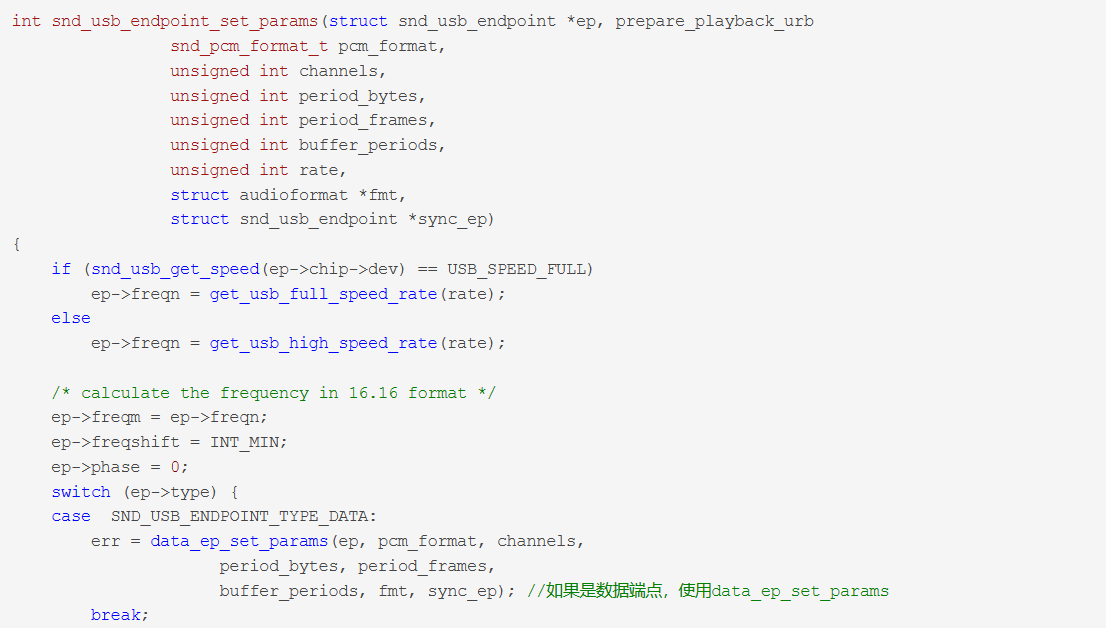
应用调用ioctl： ioctl(pcm->fd, SNDRV\_PCM\_IOCTL\_PREPARE)  //主要是传输前的准备

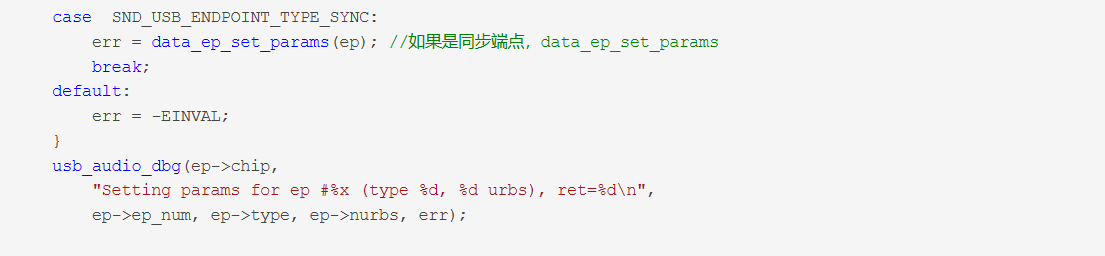
对于UAC设备通过一系列掉用就会到snd\_usb\_pcm\_prepare，





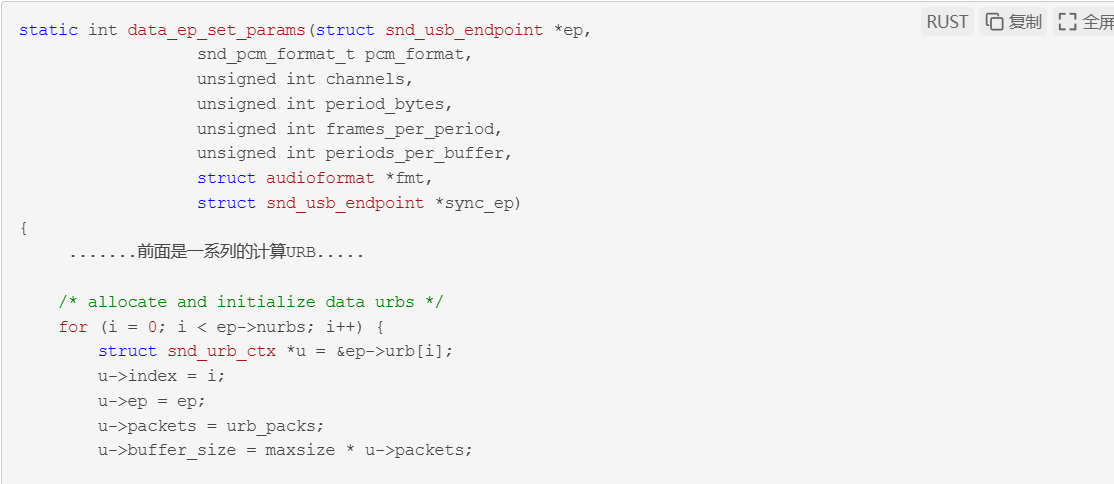
configure\_endpoint里面会调用snd\_usb\_endpoint\_set\_params设置一个snd\_usb\_endpoint。数据端点用于发送音频数据，同步端点用于接收设备端的反馈。

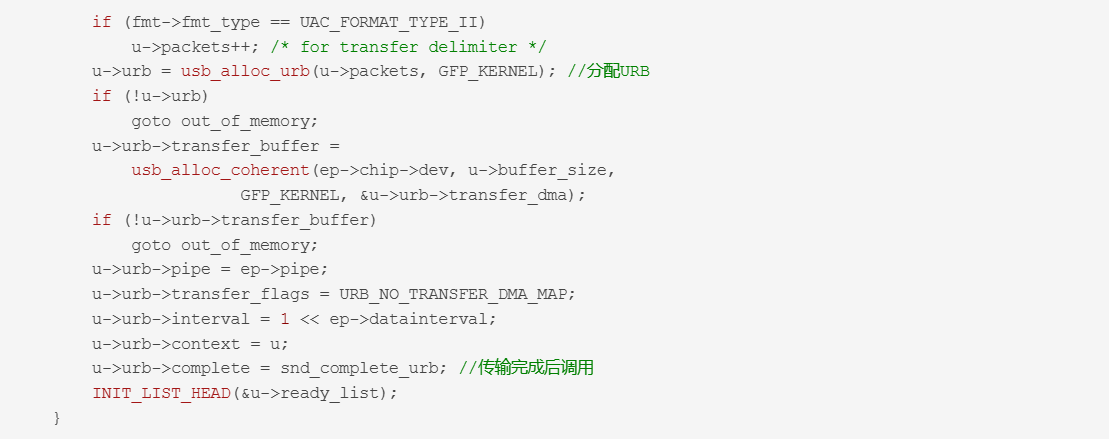




**data\_ep\_set\_params**

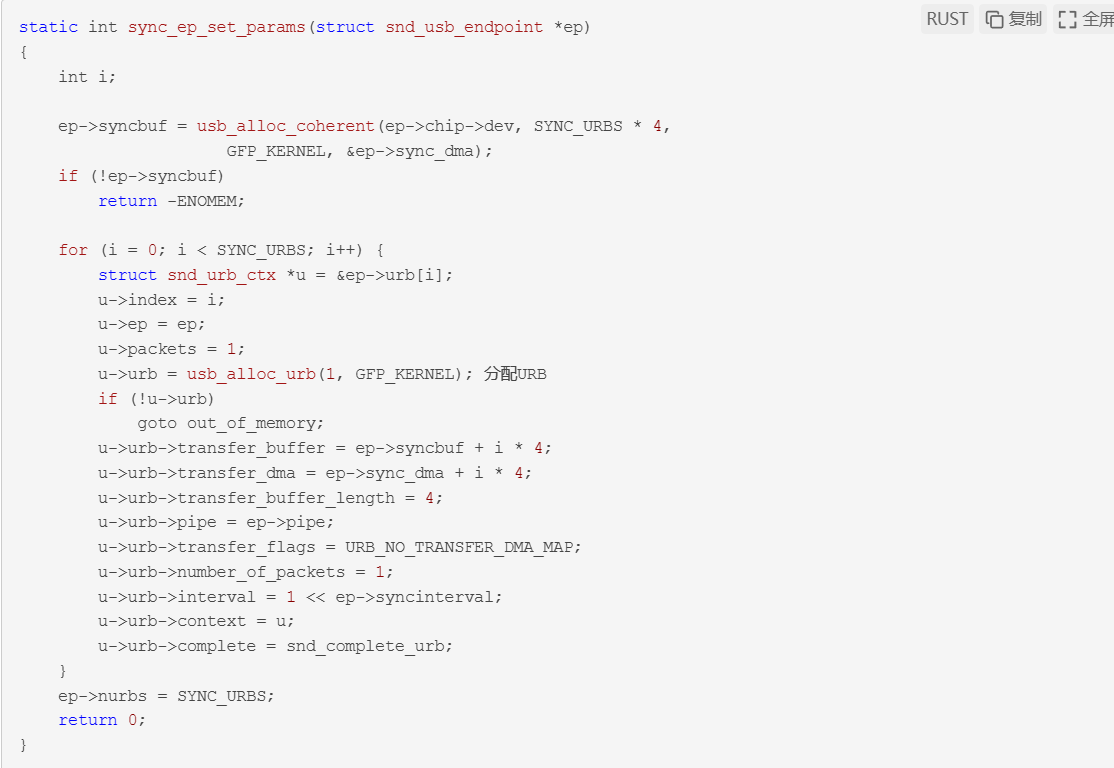
data\_ep\_set\_params里面会分配URB，传输完成后调用snd\_complete\_urb。**因为等时 urb 没有初始化函数，必须手动初始化，所以下面是包括分配和初始化了。**





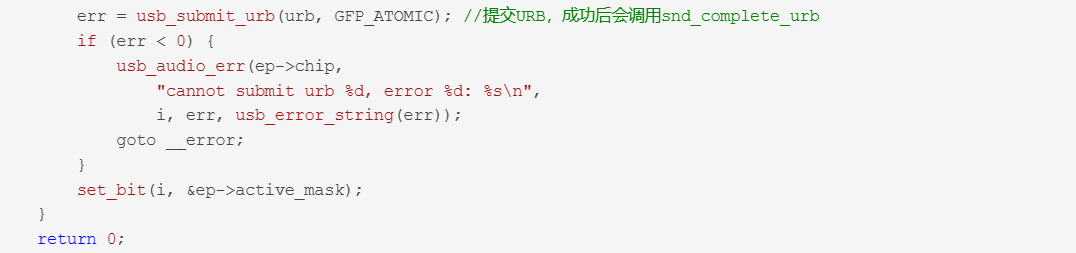
**sync\_ep\_set\_params**

里面会分配URB，传输完成后调用snd\_complete\_urb。因为等时 urb 没有初始化函数，必须手动初始化，所以下面是包括分配和初始化了。



**start\_endpoints调用snd\_usb\_endpoint\_start。**





**4.写入数据**

应用调用ioctl：ioctl(pcm->fd, SNDRV\_PCM\_IOCTL\_WRITEI\_FRAMES, &x)

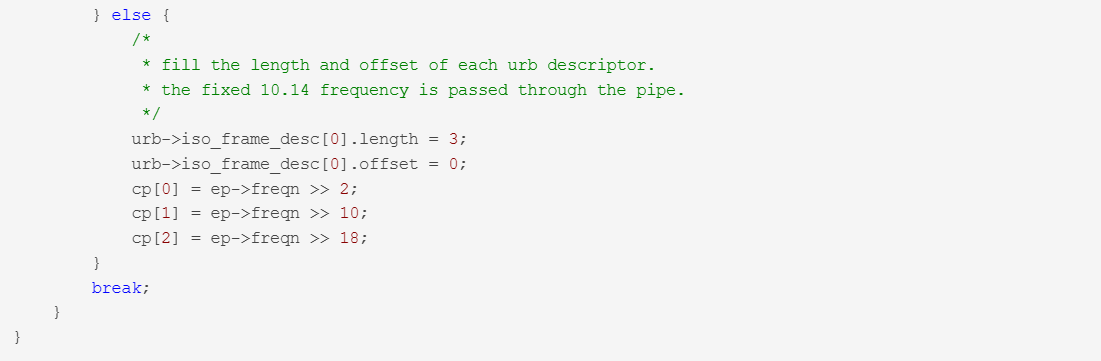
pcm\_write写下来的数据都是放在substream->runtime->dma\_area里面。

对于UAC设备通过一系列掉用就会到snd\_usb\_substream\_playback\_trigger启动传输。前面准备阶段已经提交了URB，这里同时执行SNDRV\_PCM\_TRIGGER\_START和SNDRV\_PCM\_TRIGGER\_PAUSE\_RELEASE两个分支。



为什么这里就能启动播放能呢。准备阶段数据URB发送的是静音数据，启动之后才发送正在的数据。





**5.传输回调**

**传输成功后会调用snd\_complete\_urb。**

