项目总结Phase2

# 整体进度及状况

|  |  |  |
| --- | --- | --- |
| 1 | 开发一个应用程序。用VS2010，C或C++开发一个USB声卡的声音 loop。即：打开 USB 声卡，读取声音数据，然后输出，即耳机里能听到自己 MIC 的声音。延时越小越好。 建议用 DirectShow 和线程。（提示：声音数据包的长度，是影响延时的一个因素。一般5ms一个数据包） | **[已经完成]** |
| 2 | 在windows驱动层KMDF（Kernel-Mode driver work)，写一个filter。用VS2010和WDK,，功能与第一步相同： 将上面耳机里听到MIC的声音实现，但是声音数据的传输，直接在filter完成。不通过应用程序。 | **[环境准备完成]** |

# 具体问题及处理结果

|  |  |  |
| --- | --- | --- |
| 1 | 还未找到控制数据包延迟的相关方法； | **[已经解决]** |
| 2 | 延迟测量方法精度是否能够接受； | **[已经解决]** |
| 3 | 如果精度能够接受，如何进一步降低延迟； | **[已经解决]**  **[硬件有变]** |
| 4 | 用directshow中的call back(hooker)机制。 | **[已经解决]**  **[硬件有变]** |

# 具体解决方法

## 问题1

最终改用Kernel-Streaming解决延迟。

## 问题2

经邮件交流，测量精度可以接受。

## 问题3与问题4

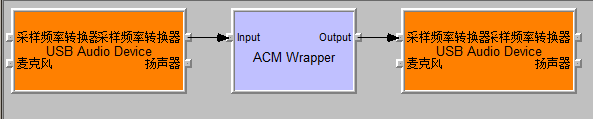
主要解决的问题就是为DirectShow回放程序延迟的问题。主要的解决方式为采用Kernel-Streaming的方式。具体思路为在DirectShow中使用WDM驱动的filter。

与一般的filter不同，一般的filter使用.dll一类的链接库与下层硬件交互，但是采用WDM驱动的filter使用ksproxy.ax直接与下层硬件交互，在真正的工作过程中，直接形成Kernel-Streaming，尽可能的减少内存上的数据交换情况。实施也表明，在使用了WDM的驱动建立了Kernel-Streaming的应用程序之后，延迟控制能控制在49ms，甚至建模级别（仅仅为DirectShow的GraphEdit层面实现）能够实现33ms的延迟。

### Kernel-Stream的GraphEdit层面实现

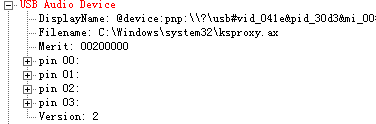
#### 分立式Kernel-Streaming实现

在使用以下的Graph之后，能够明显的感觉到延迟控制的优化，还有就是回放音质的增强：

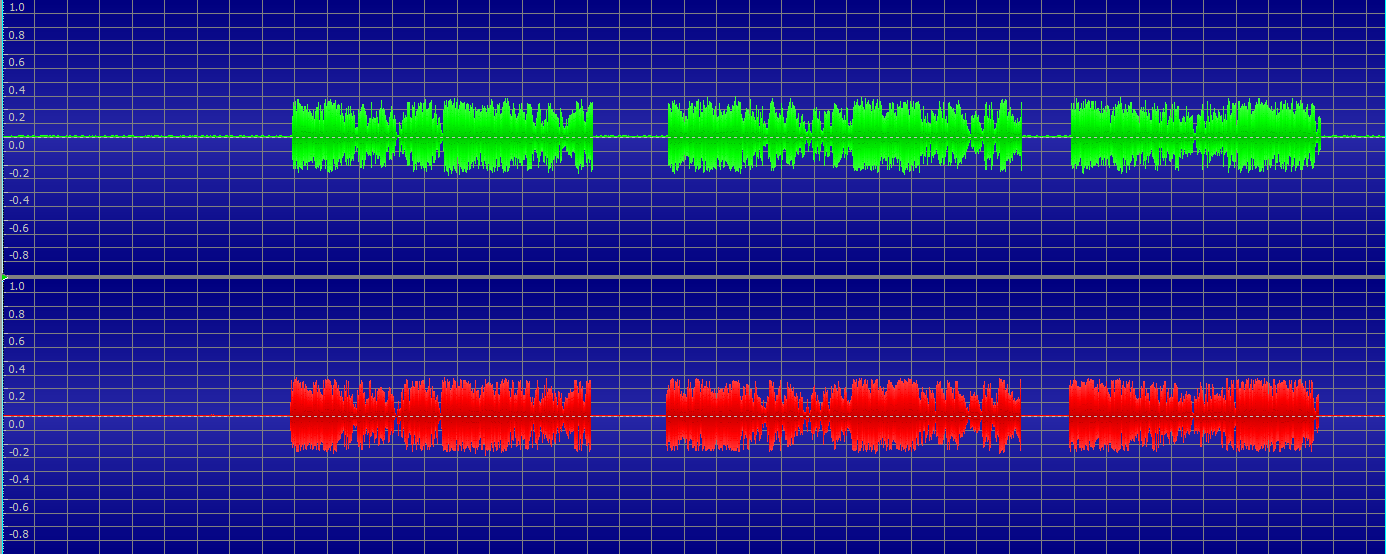


在逻辑图上进行连接之后，GraphEdit会自动添加上一个普通的Filter用于Pin指针之间的媒体格式适应。正是由于这个原因，这个图无法最终实现应用程序级别的实现。

关于Kernel Stream与一般情况下的驱动，其采用的启动点不同，而运作方式也有很大的不同。其采用的连接库为

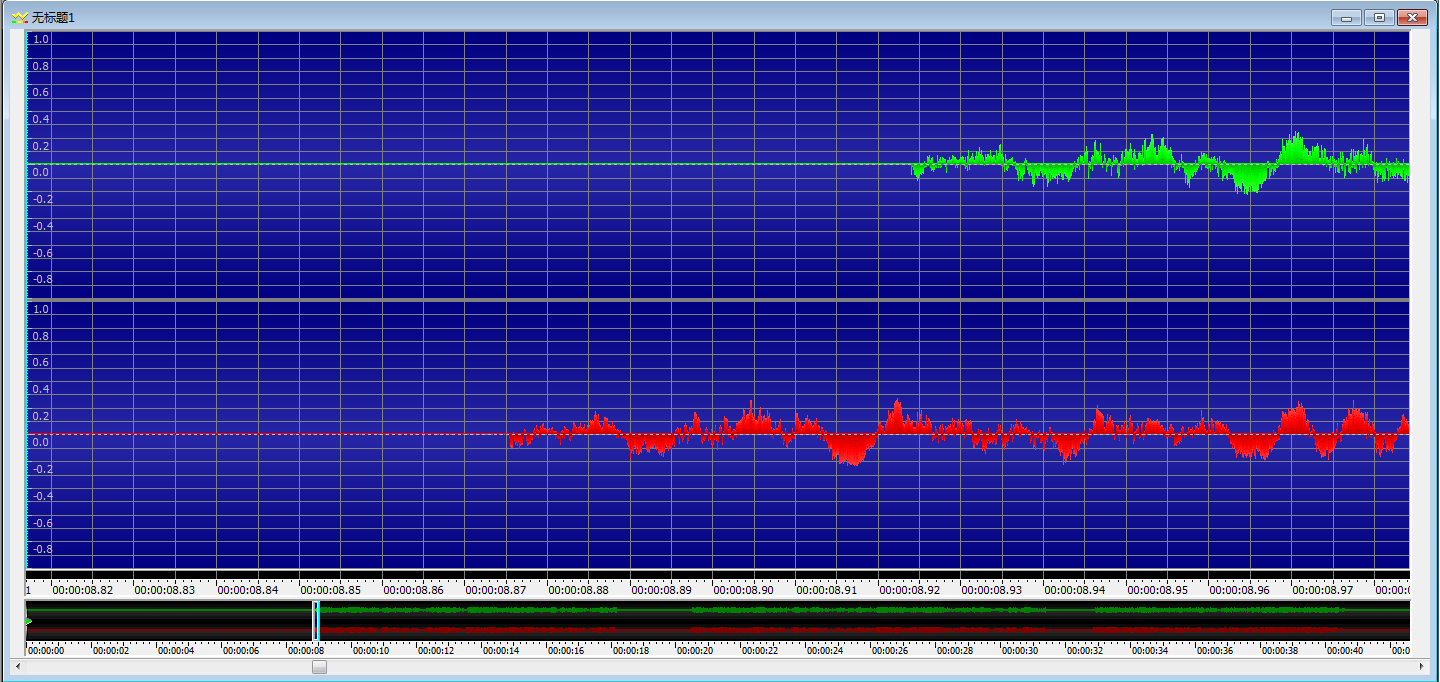


延迟表现



测试下来的波形如图，但从表现上来看，比先前进行的测试，还原度与延迟控制都更加理想一点，而且为MIC输入，并非为线路输入。

具体的延迟为。



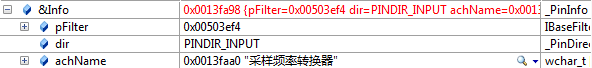
延迟已经低于100ms一下，而且并未使用缓存延迟控制。

具体的延迟水平为：

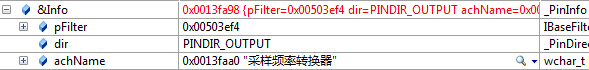


49ms  
采样文件test8.wav

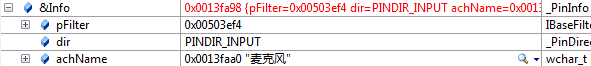
应用实现中进行了指针枚举在采用WDM Streaming Capture设备进行处理之前，首先要对设备进行接口的枚举，等找到合适的设备接口之后，才能进行有效的连接。由于内核驱动的特殊限制，只能在debug下看到接口的详细信息。



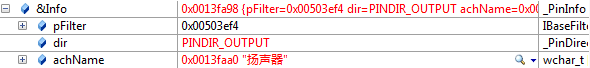








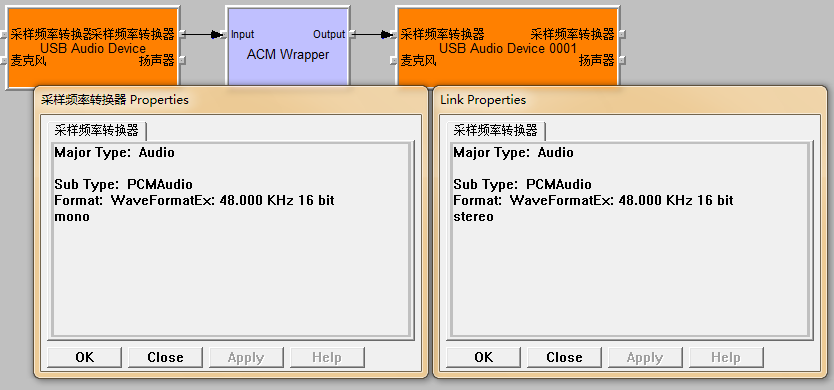






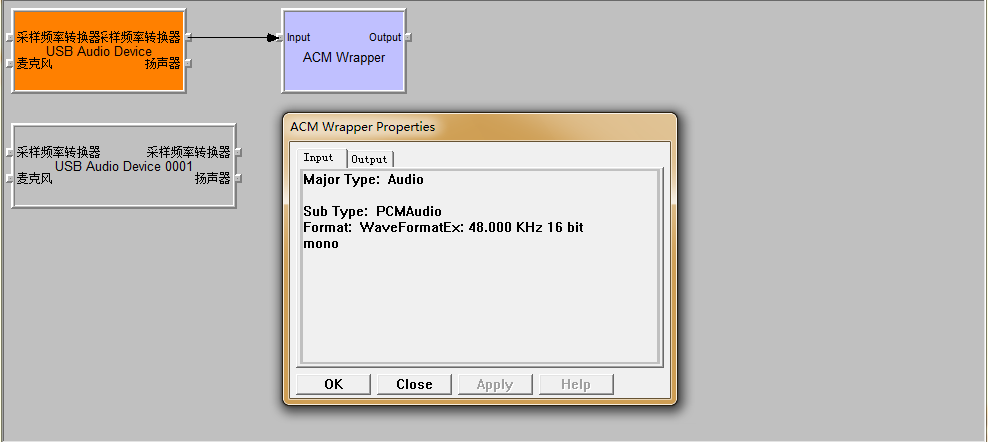
四个接口枚举完成之后的显示

但是由于自动增加了ACM Wrapper，最终也没能成功的连结出测试所用的Graph。



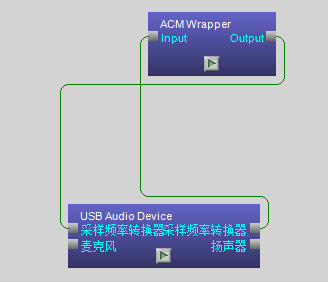
最后可以发现连接不成功的主要问题是，音频的格式不匹配。在声卡出口处为单声道，但是入口处必须为双声道。

最终在debug的时候可以发现主要的问题是最后一个线无法连接。



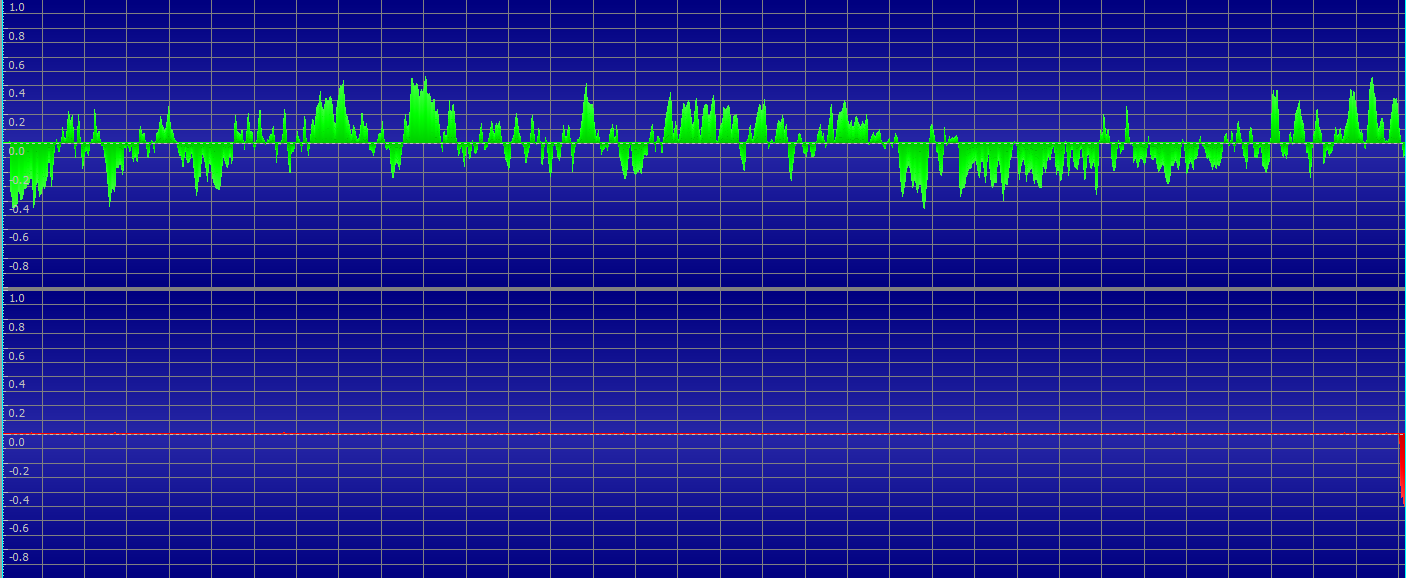
#### 独立式Kernel-Streaming实现

当使用下图的Graph进行测试的时候，延迟能够进一步得到控制，就可以实现到纯loop的Kernel-Streaming。



则延迟控制能够进一步降低至33ms。





采样文件test9.wav

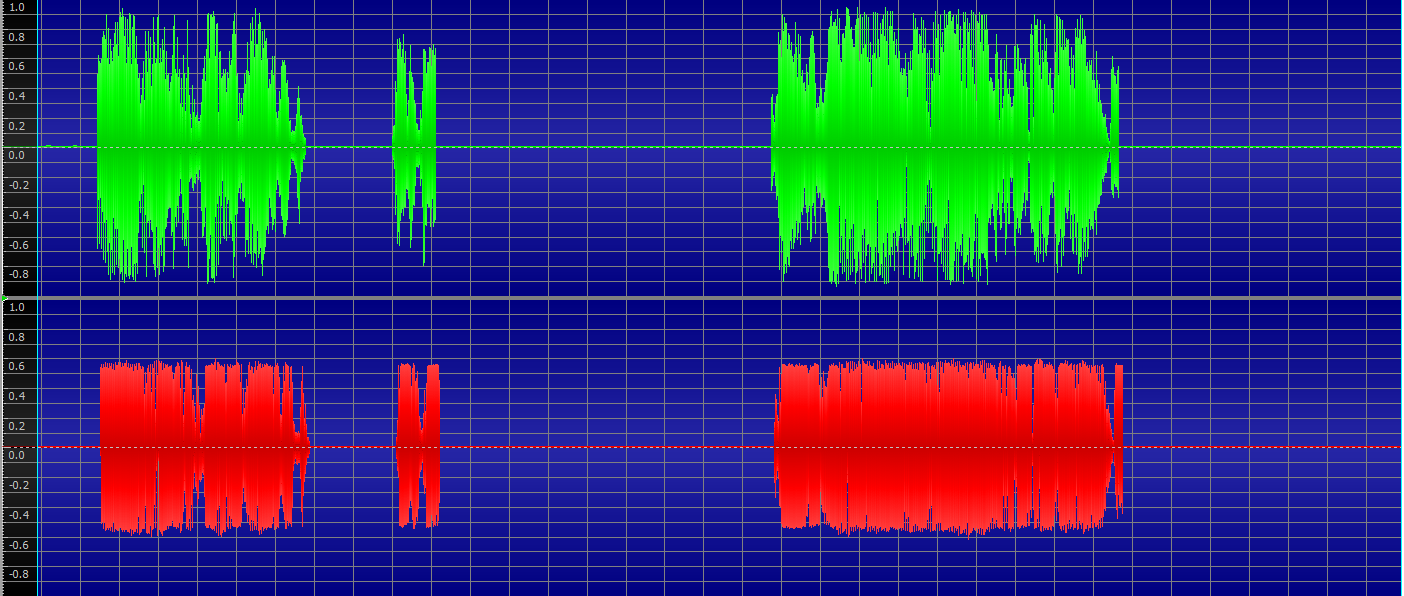
### 代码实现

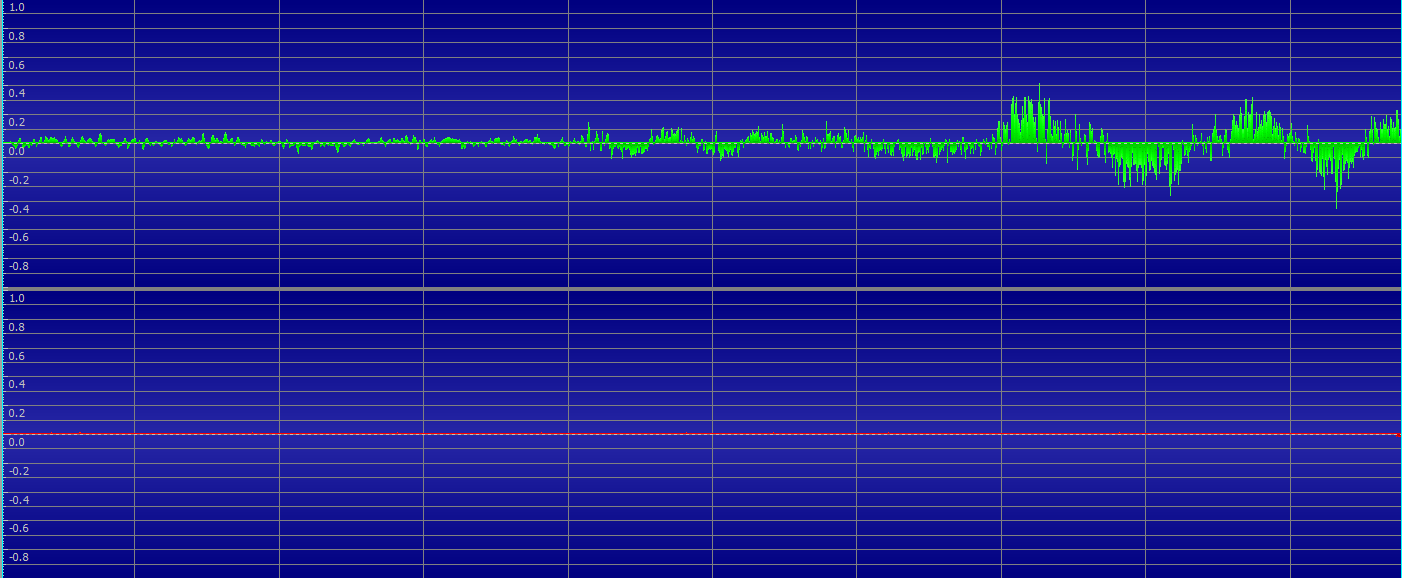
#### Kernel-Streaming直接输出

当使用如下图时能够正常进行代码实现：



测试图形如下：





延迟控制为



48ms

样本文件为test12.wav

实验表明Release和Debug版本的延迟水平相差不大。

### 项目结果

对应Git源内的proj文件夹

A1为缓存控制的可用测试板回放程序；

A2为内核流创新声卡的非可用回放程序；

A3为内核流单创新逻辑声卡的可用回放程序。

# 遗留问题

|  |  |  |
| --- | --- | --- |
| 1 | 还需不需要进一步实现独立式Kernel-Streaming的项目代码？ |  |