

转载houxiaoni012021-06-16 16:48:27137收藏 2

分类专栏:OpenMAX文章标签:openmax组件C多媒体

OpenMAX 专栏收录该内容2订阅4篇文章订阅专栏

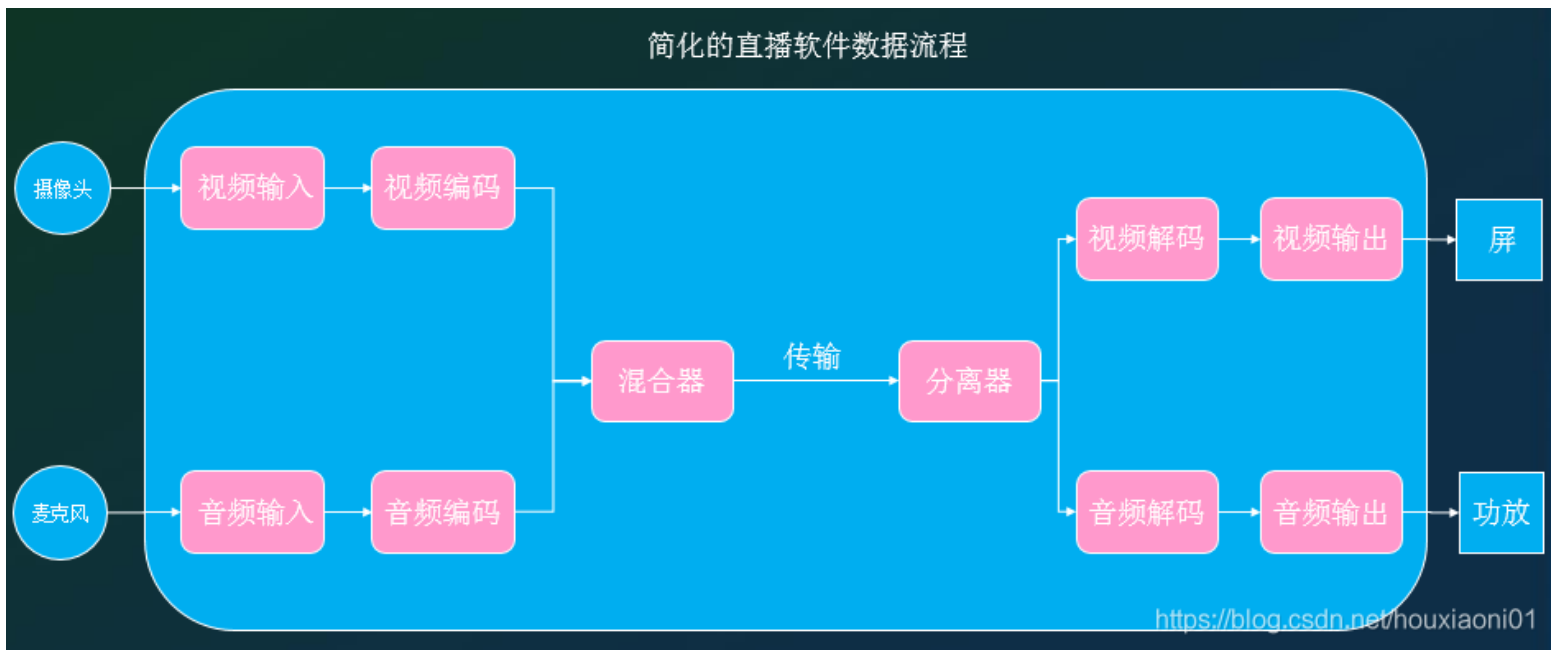
OpenMAX的重点组成部分就是组件，OpenMAX通过将meida流过程中的各个模块抽象化为组件来进行耦合，在OpenMAX标准下，数据流通过组件来进行传递、处理、显示。在该篇文章里，不需要了解细致的组件内部实现机理，也不需要知道各种方法的代码实现形式（如遇少量代码形式的解析说明可暂时略过，只需要知道该段代码要完成的工作是什么即可），通过阅读该文章，需要知道的是组件是什么？它有什么用处？内部的主要组成结构是怎样的？更加细节性的介绍放到后面的文章里面介绍。

- OpenMAX IL spec手册下载：https://www.khronos.org/files/openmax_il_spec_1_0.pdf
- OpenMAX IL sample下载：https://www.khronos.org/files/openmax/sample_implementation/OMX_CONF_MyComponent_Alt.c
- OpenMAX IL 头文件下载：https://www.khronos.org/files/openmax/headers/omx_il_v1/omx_il_v1.zip

什么是组件

组件是OpenMAX对meida视频流中的模块的抽象，比如视频输入模块、视频编码模块、视频解码模块等均可理解为XXX组件，OpenMAX提供了完整的组件式编程解决方案，包括数据流的交换与同步等。在OpenMAX IL层，组件代表着一个独立的功能性模块，组件可以是源（视频输入），目的（视频输出），编解码，滤波器，分离器（音视频分离），混合器（音视频混合）或者其它任意的数据处理模块。一个组件可以代表硬件设备、软件编解码器、处理器或者它们的组合，这取决于组件的实现方式。

一个典型的组件结构如下图所示：

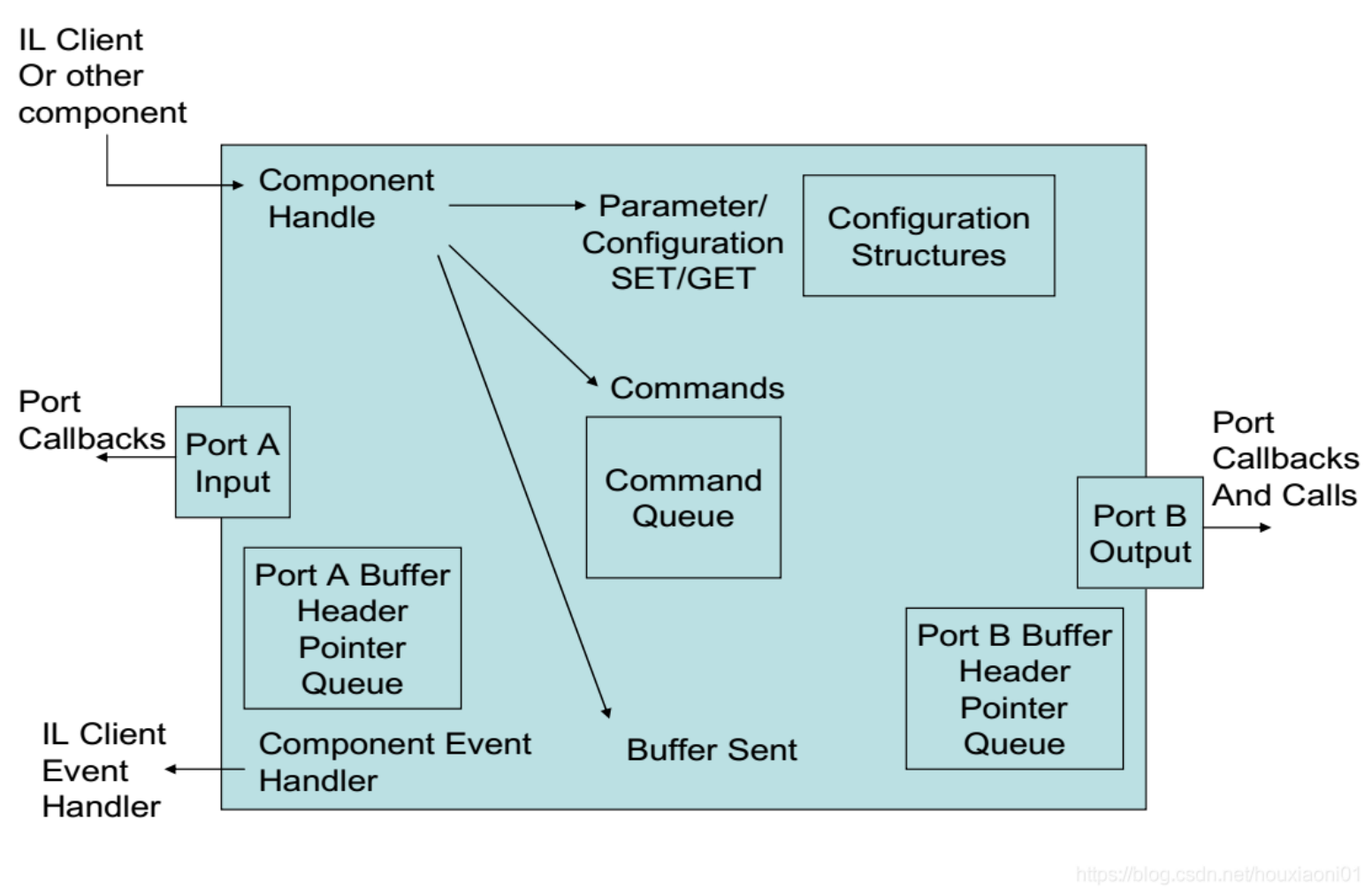


假想的直播软件组件形式

其中的视频编码、视频解码等都可以作为一个组件的形式存在。

组件的内部构造

下图是OpenMAX IL的spec手册给出的一个抽象化的组件的内部结构图：

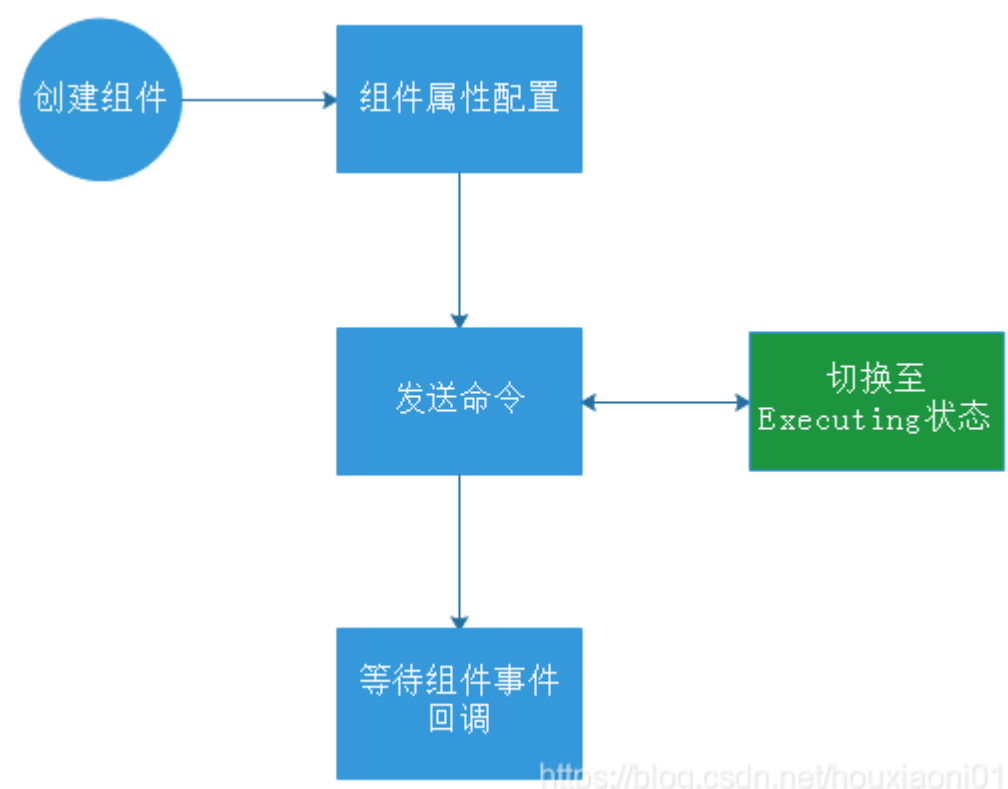


组件抽象化构造图

从上图可以看出组件内部大概有这么几个组成部分：

- 组件句柄（组件的描述符，类似文件描述符一样）。
- 配置相关的结构体方法（set/get parameter/configuration）。
- 命令队列。
- 端口（port）。
- 端口的buffer管理。
- 组件事件句柄（用于向IL Client发送事件）。
- buffer发送模块。

IL Client对组件的操作流程大概如下：



IL Client对组件的初始化流程

组件句柄

组件句柄是IL Client操作组件的信物，可以看作是跟Linux或者Windows文件编程当中的文件描述符一样的东西，在文件编程中，文件的一切操作都是根据文件描述符来完成的，组件的一切操作也是有组件句柄来完成的。可以把组件句柄当作整个组件的抽象化实例。

与配置相关的结构体方法

主要的方法有set/get parameter，set/get configuration 两类，通过这两类方法IL Client可以对组件的各个属性进行设置以及获取组件的各个属性。对于OpenMAX来说，它们两类方法的作用有所区别。**注意：这些回调方法需要自己去实现其具体的动作代码。**

- Parameter
 - Set Parameter具体到OpenMAX的相关组件代码中就是OMX_SetParameter 这个宏（该宏的详细实现在后面的文章里面会介绍），使用这个宏将会从IL Client向组件发送一个parameter 结构体，该结构体内部就包含有需要设置给组件的各种参数信息。当该宏被使用之后，组件内部的相关回调函数就会被调用，然后

houxiaoni01
码龄4年 暂无认证

38原创2万+周排名1万+总排名21万+访问等级

2099积分191粉丝169获赞31评论810收藏

私信关注

搜博主文章

热门文章

常见音频编码格式解析 47115

常见视频编码格式解析 32508

Gstreamer学习笔记（7）：plugin注册流程分析（超详细） 6917

Git学习：git 生成 patch的命令 6642

音频硬件基础 5871

分类专栏

显示相关 1篇

Alsa 7篇

OpenMAX 4篇

GStreamer 16篇

FFmpeg 1篇

流媒体 3篇

最新评论

Gstreamer学习笔记（2）：GstElement...
lhdaniu: 请问，/* 创建过滤波器元件 */ decoder = gst_element_factory_make ("mad", ...
【H264/AVC 句法和语义详解】(三): NA...
范JJ: 顶

Gstreamer学习笔记（9）：message, ev...
alexsender: mark

动态范围控制（DRC）简介
houxiaoni01: 谢谢

动态范围控制（DRC）简介
houxiaoni01: 谢谢，边学习边记录的

您愿意向朋友推荐“博客详情页”吗？

强烈不推荐 不推荐 一般般 推荐 强烈推荐

最新文章

TWI-I2C学习详解

【I2C】i2c-tools的使用方法

linux内核调试技巧 dump_stack()

2021年 23篇2020年 24篇

2019年 76篇2018年 19篇

2017年 6篇

目录

什么是组件

组件的内部构造

- 组件句柄
- 与配置相关的结构体方法
- 命令队列
- 端口
- 端口的buffer管理
- 组件的事件句柄
- buffer发送模块
 - tunnel模式（绑定模式）
 - Non-tunnel模式（非绑定模式）
- 组件的状态转换





- Get Parameter
这个就不必详细解释了，自然是获取组件的参数，具体到OpenMAX的相关组件代码中就是 **OMX_GetParameter** 这个宏。
- Configuration
 - Set Configuration
具体到OpenMAX的相关代码中就是 **OMX_SetConfig** 这个宏（该宏的详细实现在后面的文章里面会介绍），使用该宏将会设置组件的相关配置的值。该宏可以在组件初始化并且状态转为 **Loaded** 之后的任意时刻调用。调用这需要提供配置结构体的内存地址以及已经初始化过的结构体内部相关成员，在该宏被使用完毕后，IL Client可以丢弃相关的配置结构体（组件内部会复制一份相关的配置）。比如可以设置组件的时间戳或者是时间跳转单位以及时间缩放系数等。
 - Get Configuration
道理同上，只不过该方法是用来获取组件配置的，具体到OpenMAX的相关代码中就是 **OMX_GetConfig** 这个宏。
- 宏定义代码实现（不必过于纠结其进一步的实现细节，留着后面讲解）

```
1  /* get/set parameter宏定义的实现 */
2  #define OMX_GetParameter(
3      hComponent,
4      nParamIndex,
5      pComponentParameterStructure)
6      ((OMX_COMPONENTTYPE*)hComponent)->GetParameter(
7          hComponent,
8          nParamIndex,
9          pComponentParameterStructure) /* Macro End */
10 #define OMX_SetParameter(
11     hComponent,
12     nParamIndex,
13     pComponentParameterStructure)
14     ((OMX_COMPONENTTYPE*)hComponent)->SetParameter(
15         hComponent,
16         nParamIndex,
17         pComponentParameterStructure) /* Macro End */
18
19 /* get/set configuration宏定义的实现 */
20 #define OMX_GetConfig(
21     hComponent,
22     nConfigIndex,
23     pComponentConfigStructure)
24     ((OMX_COMPONENTTYPE*)hComponent)->GetConfig(
25         hComponent,
26         nConfigIndex,
27         pComponentConfigStructure) /* Macro End */
28 #define OMX_SetConfig(
29     hComponent,
30     nConfigIndex,
31     pComponentConfigStructure)
32     ((OMX_COMPONENTTYPE*)hComponent)->SetConfig(
33         hComponent,
34         nConfigIndex,
35         pComponentConfigStructure) /* Macro End */
```

命令队列

命令队列用于IL Client与组件的各种控制，比如下面这么几种命令（**控制命令对应的动作代码需要自行实现**）：

command	function
OMX_CommandStateSet	设置组件的状态
OMX_CommandFlush	冲洗相关端口的buffer
OMX_CommandPortDisable	停止指定的端口
OMX_CommandPortEnable	使能指定的端口
OMX_CommandMarkBuffer	标记一个buffer，指定哪个组件将响应这个事件

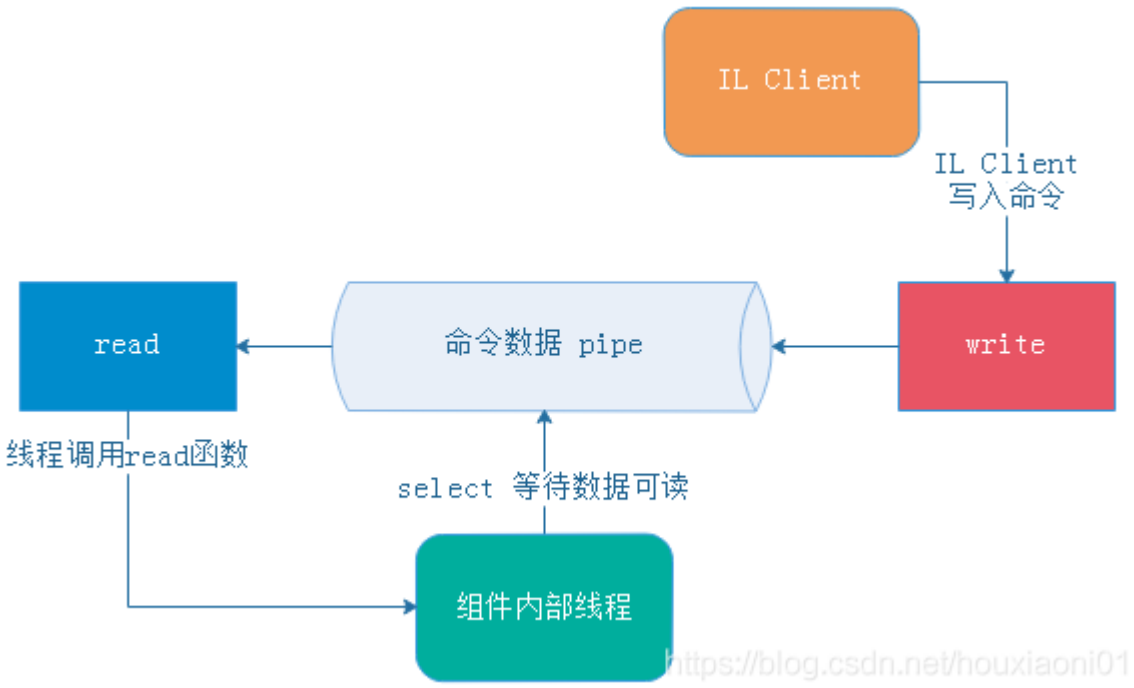
组件部分命令

在OpenMAX的sample里面，命令队列是采用pipe的形式实现的，具体分为：

1. 在组件的初始化代码里面新建一个pipe（使用 **pipe** 函数），此时就可以获取到一个类似于文件句柄一样的pipe实例。
2. 在组件内部线程里面等待pipe可读（有命令到达），使用 **select** 函数等待，完全可以将上一步生成的pipe实例当作文件描述符（文件句柄）来用。
3. 在相关的回调函数里面（对于命令的发送来说，宏定义为 **OMX_SendCommand**）往pipe里面写命令数据，使用write函数，参数就是第一步获取的pipe实例以及命令结构体。
4. 组件内部线程的 **select** 检测到pipe可读，则立马读出一个命令进行执行，执行完毕之后再次转到步骤2进行循环。

暂时不需要知道组件命令队列的代码实现细节，也不需要知道命令的产生过程，只需要了解命令的大致作用即可

除了上面的用 **pipe** 结合 **read**，**write** 的形式进行命令队列的管理之外，还可以自行构建一个生产者-消费者的数据队列模型进行命令队列的管理。其实上面那种方法就可以看作是一个生产者-消费者模型，其工作流程如下图所示：



命令pipe

上图中的组件内部线程即可当作消费者，而IL Client则可以当作生产者，生产者不断地往命令pipe数据库里面写入命令（命令的生产），消费者不断地从命令pipe里面读取命令（消费命令），两者异步进行，互不干扰，并且命令也不会由于系统繁忙而丢失。

还有一种自行实现的生产者-消费者模型，是通过链表来实现的，其基本流程如下图所示：



houxiaoni01

码龄4年 暂无认证

38

2万+

1万+

21万+

原创

周排名

总排名

访问

等级

2099

191

169

31

810

积分

粉丝

获赞

评论

收藏

私信

关注

搜索主文章

Q

热门文章

常见音频编码格式解析 47115

常见视频编码格式解析 32508

Gstreamer学习笔记（7）：plugin注册流程分析（超详细） 6917

Git学习：git 生成 patch的命令 6642

音频硬件基础 5871

分类专栏

显示相关

1篇

Alsa

7篇

OpenMAX

4篇

GStreamer

16篇

FFmpeg

1篇

流媒体

3篇

最新评论

Gstreamer学习笔记（2）：GstElement...
lhdaniu: 请问，/* 创建过滤器元件 */ decod
er = gst_element_factory_make ("mad", ...
【H264/AVC 句法和语义详解】(三): NA...
范JJ: 顶

Gstreamer学习笔记（9）：message, ev...
alexsender: mark

动态范围控制（DRC）简介
houxiaoni01: 谢谢

动态范围控制（DRC）简介
houxiaoni01: 谢谢，边学习边记录的

您愿意向朋友推荐“博客详情页”吗？

强烈不推荐

不推荐

一般般

推荐

强烈推荐

最新文章

TWI-I2C学习详解

【I2C】i2c-tools的使用方法

linux内核调试技巧 dump_stack()

2021年 23篇

2020年 24篇

2019年 76篇

2018年 19篇

2017年 6篇

目录

什么是组件

组件的内部构造

- 组件句柄
- 与配置相关的结构体方法

命令队列

端口

- 端口的buffer管理

组件的事件句柄

buffer发送模块

- tunnel模式（绑定模式）
- Non-tunnel模式（非绑定模式）

组件的状态转换



houxiaoni01

码龄4年

暂无认证

38

2万+

1万+

21万+

原创

周排名

总排名

访问

等级

2099

191

169

31

810

积分

粉丝

获赞

评论

收藏

私信

关注

搜博主文章

热门文章

常见音频编码格式解析 47115

常见视频编码格式解析 32508

Gstreamer学习笔记 (7) : plugin注册流程分析 (超详细) 6917

Git学习: git 生成 patch的命令 6642

音频硬件基础 5871

分类专栏

显示相关

1篇

Alsa

7篇

OpenMAX

4篇

GStreamer

16篇

FFmpeg

1篇

流媒体

3篇

最新评论

Gstreamer学习笔记 (2) : GstElement...

Ihdaniu: 请问, /* 创建过滤器元件 */ decod er = gs_t_element_factory_make ("mad", ...

【H264/AVC 句法和语义详解】(三): NA...

范JJ: 顶

Gstreamer学习笔记 (9) : message, ev...

alexsendar: mark

动态范围控制 (DRC) 简介

houxiaoni01: 谢谢

动态范围控制 (DRC) 简介

houxiaoni01: 谢谢, 边学习边记录的

您愿意向朋友推荐“博客详情页”吗？

强烈不推荐

不推荐

一般般

推荐

强烈推荐

最新文章

TWI-I2C学习详解

【I2C】i2c-tools的使用方法

linux内核调试技巧 dump_stack()

2021年 23篇

2020年 24篇

2019年 76篇

2018年 19篇

2017年 6篇

目录

什么是组件

组件的内部构造

组件句柄

与配置相关的结构体方法

命令队列

端口

端口的buffer管理

组件的事件句柄

buffer发送模块

tunnel模式 (绑定模式)

Non-tunnel模式 (非绑定模式)

组件的状态转换

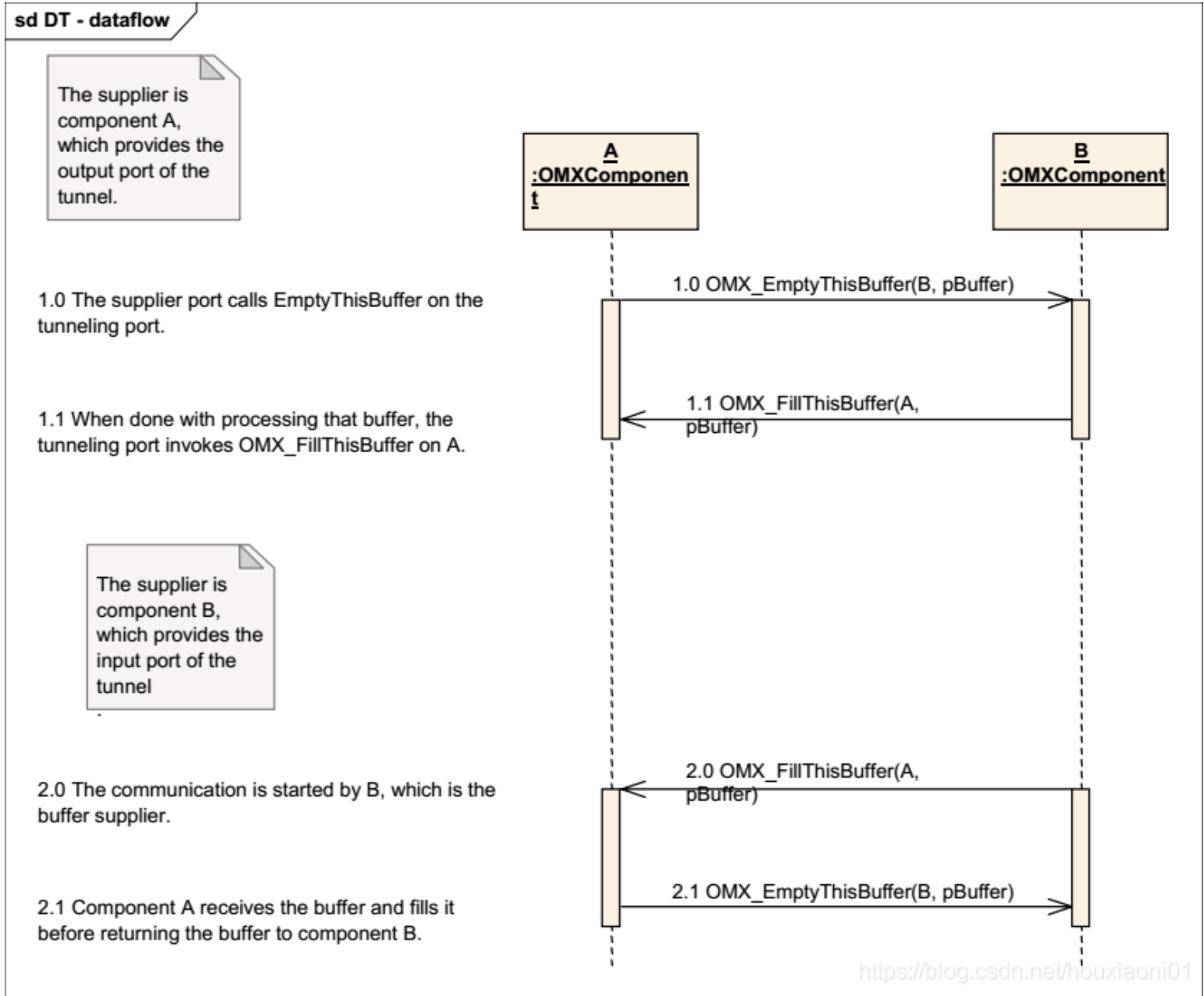
```
22     Event that the component wants to notify the application about.
23     @param nData1
24         nData will be the OMX_ERRORTYPE for an error event and will be
25         an OMX_COMMANDTYPE for a command complete event and OMX_INDEXTYPE for a
26         OMX_PortSettingsChanged event.
27     @param nData2
28         nData2 will hold further information related to the event. Can be OMX_S
29         a OMX_CommandStateSet command or port index for a OMX_PortSettingsChang
30         Default value is 0 if not used. )
31     @param pEventData
32         Pointer to additional event-specific data (see spec for meaning).
33     */
34
35     OMX_ERRORTYPE (*EventHandler)(
36         OMX_IN OMX_HANDLETYPE hComponent,
37         OMX_IN OMX_PTR pAppData,
38         OMX_IN OMX_EVENTTYPE eEvent,
39         OMX_IN OMX_U32 nData1,
40         OMX_IN OMX_U32 nData2,
41         OMX_IN OMX_PTR pEventData);
42
43     /** The EmptyBufferDone method is used to return emptied buffers from an
44     input port back to the application for reuse. This is a blocking call
45     so the application should not attempt to refill the buffers during this
46     call, but should queue them and refill them in another thread. There
47     is no error return, so the application shall handle any errors generated
48     internally.
49
50     The application should return from this call within 5 msec.
51
52     @param hComponent
53         handle of the component to access. This is the component
54         handle returned by the call to the GetHandle function.
55     @param pAppData
56         pointer to an application defined value that was provided in the
57         pAppData parameter to the OMX_GetHandle method for the component.
58         This application defined value is provided so that the application
59         can have a component specific context when receiving the callback.
60     @param pBuffer
61         pointer to an OMX_BUFFERHEADERTYPE structure allocated with UseBuffer
62         or AllocateBuffer indicating the buffer that was emptied.
63     @ingroup buf
64     */
65     OMX_ERRORTYPE (*EmptyBufferDone)(
66         OMX_IN OMX_HANDLETYPE hComponent,
67         OMX_IN OMX_PTR pAppData,
68         OMX_IN OMX_BUFFERHEADERTYPE* pBuffer);
69
70     /** The FillBufferDone method is used to return filled buffers from an
71     output port back to the application for emptying and then reuse.
72     This is a blocking call so the application should not attempt to
73     empty the buffers during this call, but should queue the buffers
74     and empty them in another thread. There is no error return, so
75     the application shall handle any errors generated internally. The
76     application shall also update the buffer header to indicate the
77     number of bytes placed into the buffer.
78
79     The application should return from this call within 5 msec.
80
81     @param hComponent
82         handle of the component to access. This is the component
83         handle returned by the call to the GetHandle function.
84     @param pAppData
85         pointer to an application defined value that was provided in the
86         pAppData parameter to the OMX_GetHandle method for the component.
87         This application defined value is provided so that the application
88         can have a component specific context when receiving the callback.
89     @param pBuffer
90         pointer to an OMX_BUFFERHEADERTYPE structure allocated with UseBuffer
91         or AllocateBuffer indicating the buffer that was filled.
92     @ingroup buf
93     */
94     OMX_ERRORTYPE (*FillBufferDone)(
95         OMX_OUT OMX_HANDLETYPE hComponent,
96         OMX_OUT OMX_PTR pAppData,
97         OMX_OUT OMX_BUFFERHEADERTYPE* pBuffer);
98 }
```

里面的三个回调成员由IL Client实现，在组件内部可以通过某种事件触发这三种回调成员的回调，比如 **EventHandler** 回调成员就会在组件状态转换完成的时候被回调，其余两个则会在 **EmptyBuffer** , **FillThisBuffer** 操作完成的时候被调用，利用这几个回调函数，IL Client可以接收来自于组件内部的若干消息事件，从而达到某种同步以及接收组件的反馈。

buffer发送模块

tunnel模式 (绑定模式)

该模块的实现是组件内部比较重要的一部分，通常位于组件内部线程的命令分发之后 (tunnel模式-也即绑定模式)，可以参考下OpenMAX sample的线程实现部分。实际的buffer发送是通过PORT端口来完成的，通过PORT端口来调用与之绑定的组件的**EmptyThisBuffer**方法来完成数据的传递。我们来看一张图：



tunnel模式下的数据传递
由图中可以看出，如果组件A是数据的提供者，那么完整的一帧数据传递就是：

- 组件A调用组件B (通过PORT端口实现) 的 **OMX_EmptyThisBuffer(B, pBuffer)** 宏来实现数据从A传递到B。
- 组件B调用组件A (通过PORT端口实现) 的 **OMX_FillThisBuffer(A, pBuffer)** 宏来完成数据从B到A的还回过程。

如果组件B是数据的提供者：

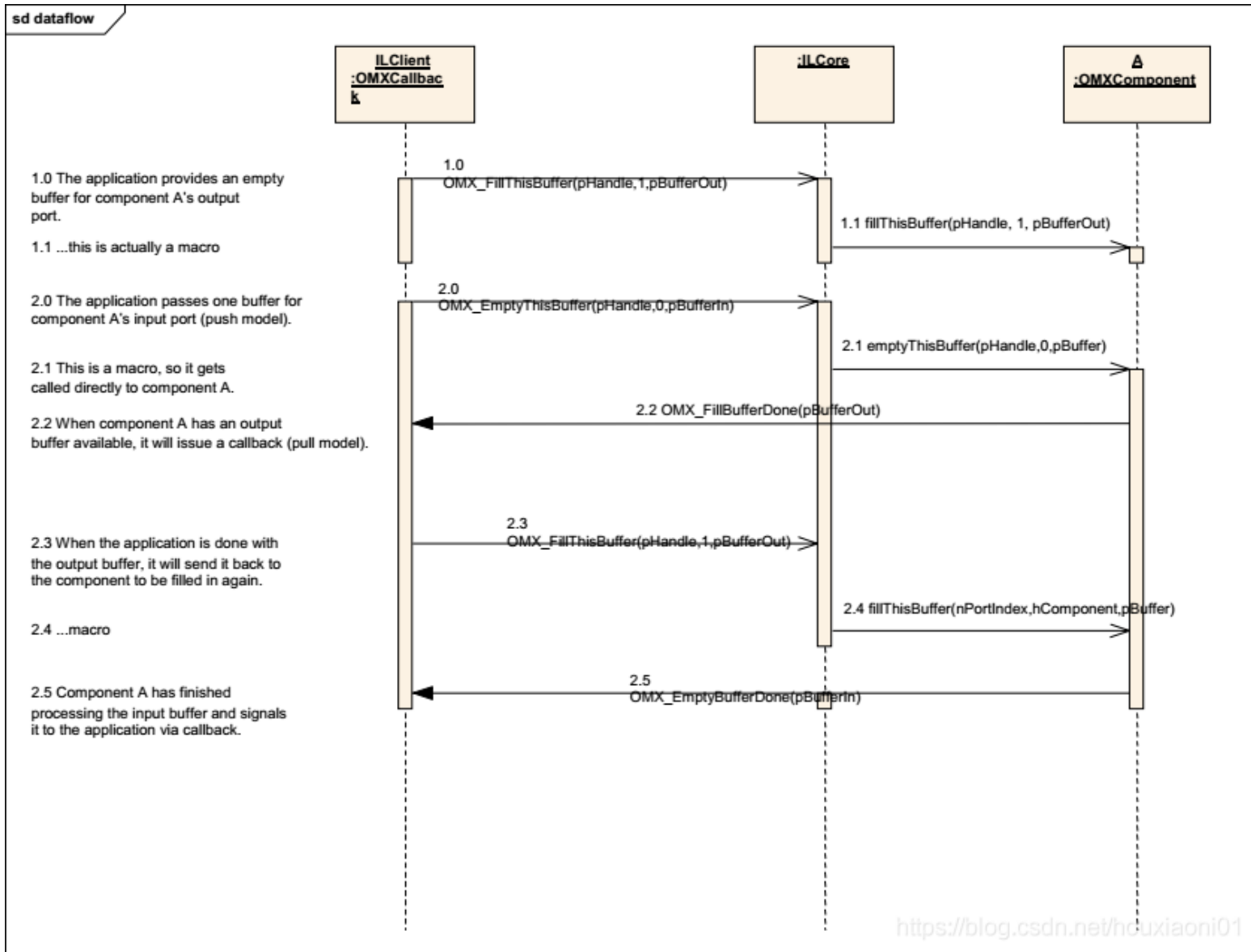


2. 组件A填充收到的buffer数据结构体，然后调用组件B（通过PORT端口实现）的 `OMX_EmptyThisBuffer(B, pBuffer)` 宏来实现数据从A到B的还回过程。

上述过程在双方组件的内部线程里面完成，至于在线程内部的具体哪个位置则取决于组件的功能以及其具体的实现方式，套路不是固定死的，可以根据自己的需求来做做出一些改变。

注意：cedarc中的openmax只实现了上面组件A作为数据提供者的情况。

Non-tunnel模式（非绑定模式）



Non-tunnel模式下的数据传递
在该模式下，数据传递的双方变为IL Client与组件了，整个过程变成了：

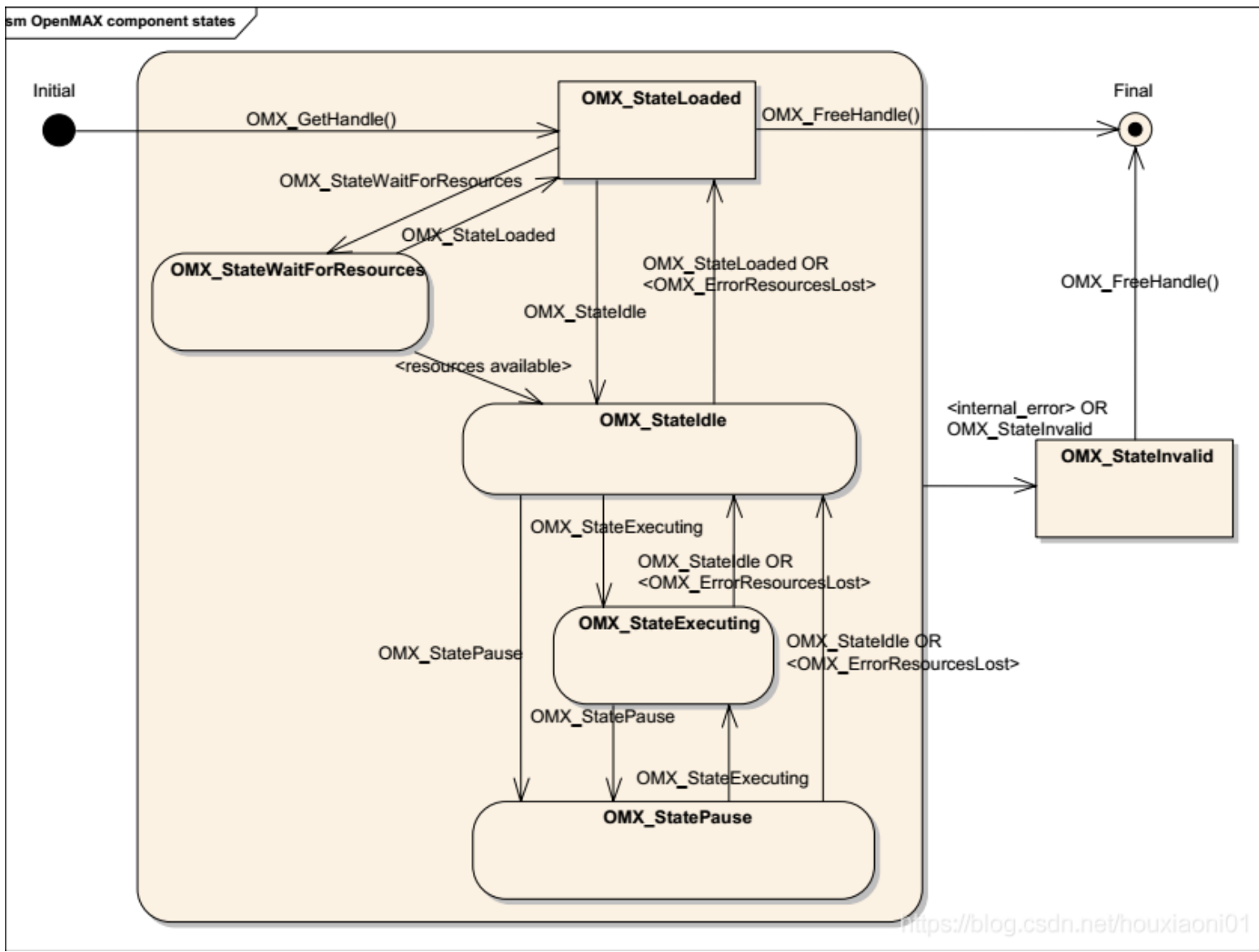
1. IL Client通过 `OMX_FillThisBuffer(pHandle,1,pBufferOut)` 回调方法向组件A的Output端口提供一个空的buffer结构体以待填充个，该宏是通过IL Core来最终完成对A组件的 `FillThisBuffer` 回调方法的调用的。
2. 然后IL Client向组件A传递一个buffer数据，这时组件A可以对buffer进行处理，然后处理完毕之后生成的新的buffer数据填充到上一步接收到的空buffer结构体里。
3. 等待组件A将Output里的buffer填充完毕，组件A就会调用 `OMX_FillBufferDone(pBufferOut)` 方法来通知IL Client接收处理后的数据。
4. IL Client数据处理完毕之后就会再次调用 `OMX_FillThisBuffer(pHandle,1,pBufferOut)` 来还回buffer到Output列表等待再次填充。
5. 组件A处理完接收到的buffer数据之后就会调用 `OMX_EmptyBufferDone(pBufferIn)` 方法来通知IL Client，说明组件A已完成接收buffer的数据处理。

上面的过程描述可能看起来有点懵，带入实例来说明下，比如组件A是一个解码组件，那么整个过程就变成了：

1. IL Client向组件A（解码组件）的Output buffer列表提供一个空的buffer结构体来存放解码后的数据。注意，此时传递给A的只是buffer的头部描述（参见上面的buffer头部描述内容介绍-端口的buffer管理），真正的buffer数据存放地址是在IL Client那里的。
2. IL Client向组件A的Input端口发送一帧待解码的buffer数据。
3. 组件A解码完毕，解码后的数据被拷贝到了Output端口buffer队列里面（根据buffer描述直接将解码后数据拷贝到IL Client的实际buffer存放地址），同时通过回调通知IL Client，buffer可用了（解码数据可用）。
4. IL Client对解码后的数据进行处理，比如保存为文件等等，然后把buffer重新还给组件A以待填充下一帧解码数据。
5. 组件A通过回调通知IL Client该帧数据已经解码完毕。可以进行下一帧数据的传递。

组件的状态转换

1. 为什么要有状态转换？
通过组件的状态转换可以控制组件之间数据传递的暂定、开始、恢复等，可用于组件间的数据同步以及数据交换开关。
2. 如何控制组件的状态？
通过组件内部提供的回调函数进行控制，主要是通过 `SendCommand` 回调来发送 `OMX_CommandStateSet` 命令来完成。
3. 都有哪些状态？
一共6种状态，类型以及状态特征和它们之间的转换关系见下图：



组件的状态转换

可以看到，组件一开始初始化的时候，状态就被初始化为 `OMX_StateLoaded`，然后等到资源准备完毕，IL Client会将组件的状态设置为 `OMX_StateIdle`，在需要传递数据的时候需要将状态设置为 `OMX_StateExecuting`，组件停止到销毁的过程正好与上面的相反。

 houxiaoni01
码龄4年 暂无认证

38原创

2万+周排名

1万+总排名

21万+访问

等级

2099积分

191粉丝

169获赞

31评论

810收藏



私信

关注

搜博文文章

Q

- 热门文章
- 常见音频编码格式解析 47115
- 常见视频编码格式解析 32508
- Gstreamer学习笔记（7）：plugin注册流程分析（超详细） 6917
- Git学习：git 生成 patch的命令 6642
- 音频硬件基础 5871


- 分类专栏
-  显示相关 1篇
-  Alsa 7篇
-  OpenMAX 4篇
-  GStreamer 16篇
-  FFmpeg 1篇
-  流媒体 3篇

- 最新评论
- Gstreamer学习笔记（2）：GstElement...
Ihdaniu: 请问，/* 创建过滤波器元件 */ decoder = gst_element_factory_make("mad", ...
【H264/AVC 句法和语义详解】(三): NA...
范JJ: 顶
- Gstreamer学习笔记（9）：message, ev...
alexsender: mark
- 动态范围控制（DRC）简介
houxiaoni01: 谢谢
- 动态范围控制（DRC）简介
houxiaoni01: 谢谢，边学习边记录的

您愿意向朋友推荐“博客详情页”吗？











强烈不推荐 不推荐 一般般 推荐 强烈推荐

- 最新文章
- TWI-I2C学习详解
- 【I2C】i2c-tools的使用方法
- linux内核调试技巧 dump_stack()
- 2021年 23篇
- 2020年 24篇
- 2019年 76篇
- 2018年 19篇
- 2017年 6篇

- 目录
- 什么是组件
- 组件的内部构造

组件句柄

与配置相关的结构体方法

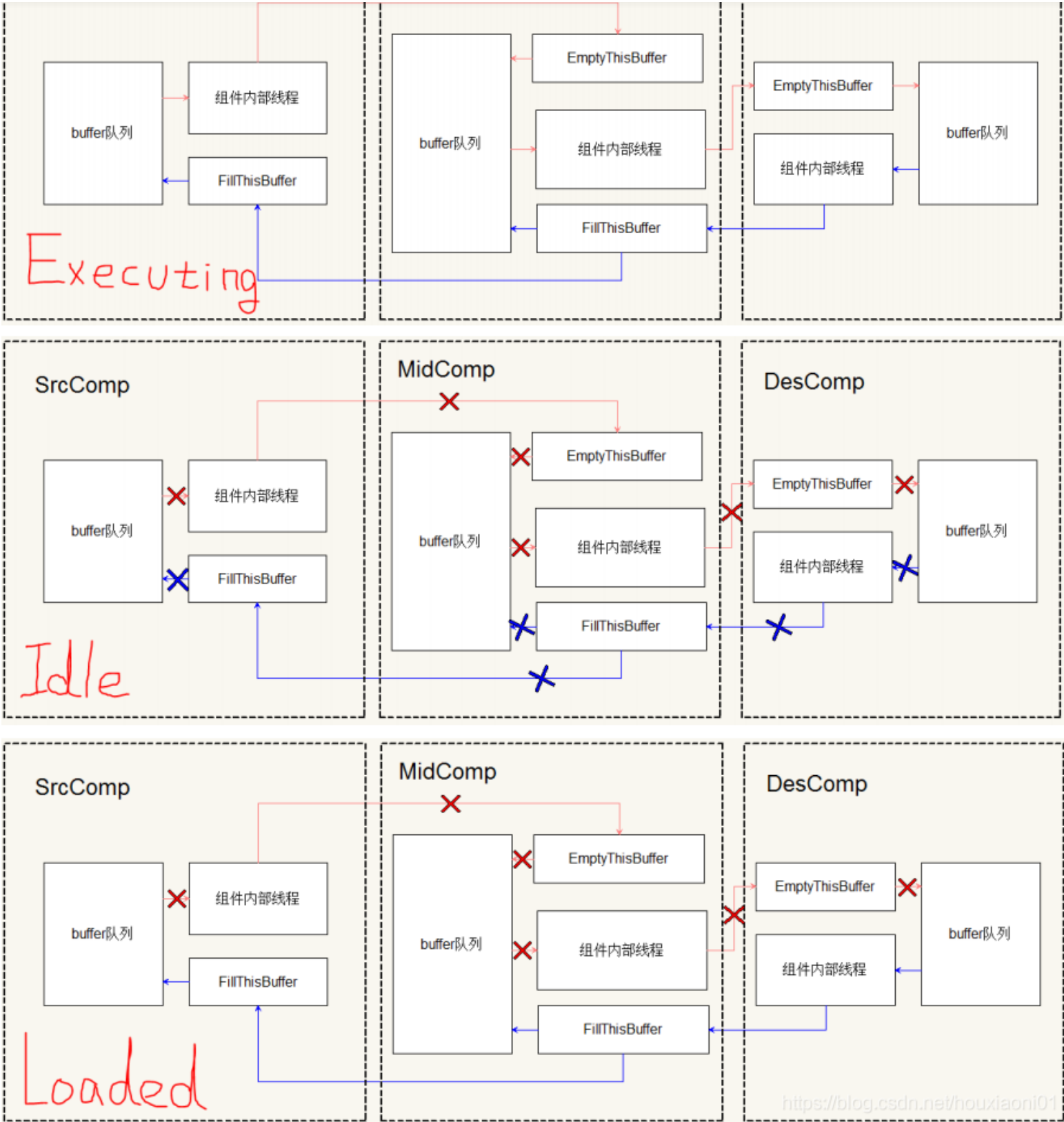
命令队列

端口

端口的buffer管理

组件的事件句柄
- buffer发送模块
- tunnel模式（绑定模式）
- Non-tunnel模式（非绑定模式）





组件状态与对应的动作

1. 组件初始化，最终状态转为Executing，数据正常流转。
2. 组件之间停止数据传输，状态转为Idle。
3. 组件的状态转为Loaded，同时后面的组件需要调用前面组件的FillThisBuffer回调方法来还回数据。
4. 组件正式销毁，所有资源释放。

本篇文章大致勾勒了一个组件的整体框架，后面介绍组件内部的相关结构体以及回调函数成员等。

文章参考：
C语言之list_head双向链表
OpenMAX编程初识

houxiaoni01 关注

👍 0 💬 1 ⭐ 2 🧑🏻 专栏目录

圈内大佬呕心之作，一年后斩获腾讯T3，这份Java学习笔记有多厉害？ 手持两把掘斤考的博主 172
俗话说“生于忧患，死于安乐”，其实大部分中年危机，就是在安乐中产生的。有的人或许会反驳，“照你这么说不，我还必须奋...

rpi-openmax-demos: Raspberry Pi的OpenMAX IL演示-源码 05-02
Raspberry Pi的OpenMAX IL演示 我想开发一个程序，该程序。 提供了可以使用RaspiCam录制视频的。 但是，这不够灵...

优质评论可以帮助作者获得更高权重

评论

amldkl: 写的好 2 月前 回复 ...

OpenMAX/IL: OMX IL 学习笔记【1】 - 结构框架_Blue_XX... 9-21
OpenMAX/IL: OMX IL 学习笔记【1】 - 结构框架 OpenMAX IL 层 API 旨在为媒体组件提供跨平台的可移植能力。这些接口...

OpenMax总结(二)OpenMax IL层结构_先行“1”步 7-21
【OpenMax IL Interface】openmax il api 是基于组件的多媒体接口,包括两大块:核心API 和 组件API。 1、core API: 用于...

raspberrypi-openmax-h264,用 树莓派 记录 H.264 视频的OpenMAX IL示例.zip 09-18
raspberrypi-openmax-h264,用 树莓派 记录 H.264 视频的OpenMAX IL示例 openmax-h264 使用 树莓派 记录 H.264 视频的...

OpenMAX编程-组件 YellowMax 1881
OpenMAX的重点组成部分就是组件，OpenMAX通过将meida流过程中的各个模块抽象化为组件来进行耦合，在OpenMAX...

OpenMax IL: component 基础知识_Seal--学海无涯 9-3
2.组件(Component):OpenMax IL的单元,每一个组件实现一种功能 3.端口(Port):组件的输入输出口 4.隧道化(Tunneled):让...

OpenMAX/IL: OMX IL 学习笔记【4】 - 实现一个组件 houxiaoni01的博客 44
1. 导读 本文聚焦于如何编程实现一个真正的组件，主题思想是介绍一个组件在编程sjd的模块组成以及如何编写，也会介绍...

OpenMAX编程-音视频等组件介绍 YellowMax 1923
本文着重介绍不同类型组件的具体构成（参数类型、特性设置等），包括audio、video、image等组件。另外对OpenMAX...

OpenMAX介绍（总括） qq69696698的专栏 1万+
一、OpenMax简介 OpenMAX是一个多媒体应用程序的标准。由NVIDIA公司和Khronos™在2006年推出。它是无授权...

OpenMax 初始化和数据流调用时序 Sailingthink的专栏 2026
OpenMax初始化和数据流的调用时序

omx core integration guide 醉月 4585
因为相关的omx spec还没有仔细阅读过，所以在这里只是对这个文档进行简单的翻译性阅读，很多概念都要在了解玩omx ...

OpenMax框架 ambitions666的博客 22
OpenMAX open media acceleration:开放多媒体加速器。跨平台的软件抽象层，用来处理多媒体。注意：这是一个标准，...

android中openMax的实现 happyzhouxiaopei的专栏 3949
一、OpenMax简介 OpenMAX是一个多媒体应用程序的标准。由NVIDIA公司和Khronos™在2006年推出。它是无授权...

OpenMAX/IL: OMX IL 学习笔记【1】 - 接口与头文件 Blue_XX的专栏 2565
1. OpenMAX IL 层的接口定义是由若干个头文件的形式给出的，在头文件中定义了一些结构体和需要开发者实现的接口函...

OpenMAX/IL: OMX IL 学习笔记【3】 - 头文件、数据结构与接口 houxiaoni01的博客 83
参考自：https://blog.csdn.net/blue_xx/article/details/46778133 1. OpenMAX IL 层头文件 OpenMAX IL 层的接口定义是以...

OpenMAX IL接口/头文件 keen_zuxwang的博客 692
OpenMAX IL接口/头文件 OpenMAX IL 层的接口定义是由若干个头文件的形式给出的，在头文件中定义了一些结构体和需...

OpenMAX flow jslywj的专栏 369
1. set callback handle and GetOMXHandle 2. SetParameter - OMX_IndexParamStandardComponentRole, 假设为 OMX_...

树莓派接收rtp h264码流解码显示 apher0的专栏 3191
修改于树莓派上的实例代码，hello_video.c 测试环境：树莓派camkit -----发送-----树莓派显示video.c 修改后如下...

树莓派raspberrypi3硬件解码H264 GPU OMX 热门推荐 Vaylb的专栏 1万+
本程序在树莓派3B上测试通过，raspberrypi3系统 话不多说，上代码： OMXH264Player.h #ifndef OMXH264PLAYER_...

opencore之OMX_Init() ccskyer的专栏 2240
opencore: omx_sharedlibrary文件将core中的API和MasterCore中的API链接起来 PVOMXInterface() { // set the...

openmax IL 学习 haima1998的专栏 1637
二、Omx架构： 二、Omx 使用： 1. initialize. OMX_Init(); 2. get componenthandle. OMX_GetHandle(); For example: ...

第十三章轴对称_小结与复习PPT课件.pptx 最新发布 10-09
第十三章轴对称_小结与复习PPT课件.pptx

