# [USB Video Class及其實現](https://sites.google.com/site/rgbbones2/usb_video_class)

[Home](https://sites.google.com/site/rgbbones2/home) > [Linux](https://sites.google.com/site/rgbbones2/linux) > [MyLinuxDocs](https://sites.google.com/site/rgbbones2/mylinuxdocs) > usb video class - 1

## 1 Video Class 基礎概念

        Usb協議中，除了通用的軟硬體電氣介面規範等，還包含了各種各樣的Class協議，用來為不同的功能定義各自的標準介面和具體的匯流排上的資料互動格式和內容。這些Class協議的數量非常多，最常見的比如支援USB 隨身碟功能的Mass Storage Class，以及通用的資料交換協議：CDC class。此外還包括Audio Class, Print Class等等。  
        理論上說，即使沒有這些Class，通過專用驅動也能夠實現各種各樣的應用功能。但是，正如Mass Storage Class的使用，使得各個廠商生產的USB 隨身碟都能通過作業系統自帶的統一的驅動程式來使用，對USB 隨身碟的普及使用起了極大的推動作用，制定其它這些Class也是為了同樣的目的。  
        Video Class 協議的目的是給USB介面的視訊裝置提供一個統一的資料交換規範。最初版本是在2003年9月才新增到USB Class規範中的，1.1的版本更是在2005年才發佈。相比之下，Mass Storage Class 早在1998年就發佈了。支援Video Class協議的多媒體晶片也是在2005年才陸續發佈。所以USB 視訊裝置目前的現狀是，在裝置一端，多數依舊還採用原先的各種包含通用USB功能的多媒體處理晶片，主機端需要安裝專用的驅動程式，基本上各個產品之間不具備相容性。甚至對於作業系統而言，也只有在XP的SP2以後，才包含了對通用的Video class協議的支援。所以即使是某些多媒體裝置（比如Logitech最新的幾款攝影機）包含了對Video Class的支援，在Win2000等作業系統上依然需要安裝驅動程式。不過，應該說使用Video Class無疑會是一個趨勢，在相應的多媒體晶片陸續投入市場後，支援Video Class的多媒體裝置應該會在一兩年內會迅速普及開來。  
        除了在硬體上通過相應的多媒體晶片支援Video Class的裝置以外，對於包含了作業系統的智慧型手機，當然也可以在手機端通過驅動程式來實現對Video Class的支援，就好像原先支援任何一種專用的USB驅動一樣。只不過資料交換的格式不是自己隨意制訂的，而是按照Video Class的規範來實現的。  
        由於目前支援Video Class的裝置還很少，所以在Linux上還沒有開放原始碼的Video Class的主機端驅動，裝置端的Video Class驅動就更沒有見到開放原始碼的程式碼了。本文在介紹USB Video Class架構的基礎上，主要是探討Linux作業系統下裝置端Video Class驅動的實現。不過在其它平台下的實現思路應該也是類似的。

## 2 USB Video Class 協議結構

### 2.1 裝置拓撲結構

        在拓撲結構上Video Class 將視訊裝置抽象為幾個主要的硬體功能模組：  
  
    Ø 輸入端點 Input Terminal  
    Ø 輸出端點 Output Terminal  
    Ø camera端點 Camera Terminal  
    Ø 選擇單元 Selector Unit  
    Ø 處理單元 Processing Unit  
    Ø 拓展單元 Extension Unit

        下圖是一幅摘自USB\_Video\_Example 1.1.pdf （[www.usb.org](http://www.usb.org/)）的拓撲結構示例圖：

[一張含有 圖表 的圖片

自動產生的描述](https://sites.google.com/site/rgbbones2/usbvideo-1.jpg)

圖1  USB Video Camera Topology Example

從sensor和另一個復合視訊裝置得到的資料流由IT 和 CT輸入，經SU選擇送PU處理，再由OT繫結到指定的USB端點。最後由USB端點與主機互動將資料傳送到host端。在實際裝置中，可能沒有其中的某些功能模組，也可能其中的幾個模組都是由同一硬體來完成的。

### 2.2 協議層次結構

        上圖中，左半部的框架組成了Video Class中的控制介面介面，右半部的框架組成了視訊流傳輸介面介面。這兩部分購成了Video Class的主要協議框架。

**2.2.1 Descriptor Layout**

        與Class相關的資訊,當然是主機端通過向裝置端獲取描述符(Descriptor)來得到的, 下圖摘自USB\_Video\_Class\_1.1.pdf , 給出了一個Video Class協議描述符應用示例的Layout。

[一張含有 圖表 的圖片

自動產生的描述](https://sites.google.com/site/rgbbones2/usbvideo-2.jpg)

圖2  Video Camera Descriptor Layout Example

         可以看到,在Descriptor Layout中，在標準描述符裡，除了Device Descriptor, Configuration Descriptor, Interface Descriptor, Endpoint Descriptor，String Descriptor以外，還有一個USB2.0 協議中後期才新加的IAD 即 Interface Association Descriptor，用來描述多個相關Interface之間的關係，在Video Class中，IAD用來描述VideoControl Interface和VideoStreaming Interface之間的關係。  
         圖中深色的部分就是Video Class 協議相關的專用描述符（Class Specific Descriptor）了。主要就是對硬體圖像採集和處理模組的物理拓撲結構和功能的描述，以及對視訊傳輸格式（包括編碼格式，位元率等等視訊圖像相關參數）的描述。  
        通過從裝置處獲得這些描述符，主機可以得知視訊裝置端的結構及其所支援的功能。而控制這些功能模組，對資料來源和資料流進行組態，則需要通過Request來完成。

### 2.3 Request

        Request是由主機向裝置端發起的功能請求，包括所有USB裝置都需要支援的Standard Device Requests 和與Class相關的Class Specific Requests ：

**2.3.1 Standard Device Requests**

        下圖列出了USB Spec中規定的標準Request

[一張含有 資料表 的圖片

自動產生的描述](https://sites.google.com/site/rgbbones2/usbvideo-3.jpg)

圖3  Standard Device Requests

        這其中，有一部分Request是由USB控制晶片在硬體一級就直接完成的應答工作，比如SET\_ADDRESS，有些則需要由軟體來做進一步的處理，比如Set\_Configuration。軟硬體的這種任務的劃分還與具體的硬體晶片相關。因為這部分是標準協議相關，本文就不詳述。

**2.3.2 Class Specific Requests**

        Class Specific Requests的資料結構Layout與標準Request是一樣的，只是內容不同而已。VideoClass的Class Specific Requests主要根據Interface分為兩類，其下又根據具體功能模組做進一步的劃分：

    Ø VideoControl Requests  
        - Camera Terminal Control Requests  
        - Selector Unit Control Requests  
        - Processing Unit Control Requests  
        - Extension Unit Control Requests  
    Ø VideoStreaming Requests  
        - Interface Control Requests

        這其中，Interface Control Requests因為是用來在主機和裝置之間協商資料互動格式和解析度，流量等資訊的，所以一般來說是必須實現的。  
        而Camera Terminal Control Requests 和 Processing Unit Control Requests中的內容，則是目前常用的即時通訊軟體如MSN / QQ 等在其視訊控制介面上整合的控制參數。  
        其中，Camera Terminal Control Requests包含了對曝光時間，曝光模式，對焦，變焦，平移等sensor獲取資料階段時的參數控制。  
        而Processing Unit Control Requests中則包含了亮度，增益，色調，對比度，白平衡等等sensor在獲取到圖像資料後，圖像處理階段的相關參數。

 [一張含有 文字 的圖片

自動產生的描述](https://sites.google.com/site/rgbbones2/usbvideo-4.jpg)

圖4  Win2000上MSN的視訊控制介面之一

        不過實際上，以上兩者的劃分在硬體層次並不是絕對的。很多參數的控制在sensor硬體等級上是同一層次的。不過，將這些功能抽象成這兩類，正如在硬體的拓撲結構上將功能模組抽象出來一樣，有利於通用化的程式設計。

## 3 USB Video Class Linux裝置端的實現

### 3.1 驅動架構

#### 3.1.1 平台及軟體基礎

        本文討論的是USB Video Class在Linux作業系統上的裝置端實現。具體是在Omap平台上，基於USB Gadget的驅動架構來實現的。  
        USB Gadget驅動分為兩層，底層是處理與USB控制晶片硬體相關的內容，對上層遮蔽了大部分硬體相關的設定，並處理了一部分標準Request和EP0相關的標準操作流程。上層則是Class相關的部分，官方的Gadget驅動中，已經包含了File Storage（USB 隨身碟功能），RNDIS（USB網路卡功能）等的支援。考慮到Video Class的資料交換過程與File Storage有很多相似的地方，所以本文在Video Class的實現中，在大的框架上仿照了File Storage驅動的架構。

#### 3.1.2 基本框架和資料流程

        在本文實現的Video Class驅動中，整體的框架基本上分為兩大部分。  
        一部分是負責處理模組的初始化過程，並負責處理Usb匯流排上的Descriptor和Requests的互動過程。包括USB匯流排上的控制和查詢包的接收，解釋，分配和應答。  
        另一部分，是在初始化過程中啟動的一個獨立的核心執行緒。負責具體的控制指令的執行和圖像資料的獲取及傳輸工作。這其中的許多操作都有可能引起睡眠，或者需要對檔案進行操作，因此必須有一個執行緒做為依託。  
        模組的流程基本上是這樣的：  
        在init函數中向Gadget底層驅動註冊VideoClass資料結構。所有的描述符都定義為全域結構變數，在模組初始化過程中，進一步完成對描述符的填充過程，啟動獨立的核心執行緒，並註冊EP0的complete回呼函數。  
在啟動的核心執行緒中打開並初始化camera裝置。將camera設定為默認的參數組態。讀取圖像資料並將資料填充到BUF裡而後提交Request到VideoStream Interface裡的BULK IN端點中。而後睡眠等待由於資料傳送完畢被喚醒或有（異常）Exception發生被喚醒。  
        如果是資料傳送完畢，則繼續讀取，填充並行送圖像資料，如果有異常發生，則轉而處理異常。  
另一方面，哪些情況會引發異常呢？主要是驅動程式與BUS匯流排互動資訊的這部分模組中，如果發生主機端重新設定Configuration，改變USB裝置組態，或者發生匯流排插拔等引起匯流排狀態變化的時候，會產生一個相應的異常，等待核心執行緒被喚醒並處理這些異常。  
        此外，在處理Requests的時候，有時候需要與camera驅動模組互動控制資訊，同樣需要操作檔案控制代碼，目前的做法是在ep0 request的回呼函數（context）中啟動一個bottom half task，在Bottom half中完成相應的控制。  
從總體結構上說，這樣做很難看，理想的話應該在上述獨立的核心執行緒中統一處理與Camera模組相關的操作，但是目前的架構，要喚醒該執行緒，只有兩個途徑，一是資料傳輸完畢或被取消，二是有匯流排狀態變化相關的異常發生。如果硬加一個異常用來處理Requests似乎也很難看，而且對架構需要有較大的調整。所以目前只好簡化的採用了前面所說的方案。  
        然後關於什麼時候打開裝置，開始往USB匯流排上放置資料，目前的處理方式也不是非常理想。目前是在模組初始化後立即獲取第一幀圖像而後等待主機端讀取，實際上，主機端可能並不馬上安排圖像資料的傳輸。但是如果在主機端需要的時候才去讀取資料的話，因為sensor獲取資料需要一段曝光時間再加上壓縮資料所需的時間，不可能立刻響應，所以在時間上肯定不能滿足開始一段的資料傳輸請求。也需要繼續仔細分析在何時啟動camera模組最為合適。

### 3.2 與Camera驅動和V4L2子系統的配合

        在linux核心中，2002年12月起發佈了V4L2 (Video For Linux Two ) 0.1版本的規範，V4l2試圖為所有和視訊相關的裝置都提供統一的介面，這其中當然也就包括了Camera裝置。  
        而USB Video Class這一部分內容恰恰與視訊裝置也是密切相關的，所以在某些平台產品的實現中，甚至是在VideoClass中直接包含了Camera的驅動程式。這樣做對於單一產品來說，可以大大簡化驅動的層次結構，應該說是處理Camera的最直接簡潔的辦法。  
        但是，在本文的實現中，考慮到Linux核心中合理的模組劃分的原則，也是為了符合Gadget驅動的其他Class實現的一貫風格，所以還是儘量使用V4L2的介面來控制Camera模組。這樣做也有利於程式碼的移植。減小不同功能模組之間的耦合性。  
        理想的方式自然是所有的與Camera相關的操作都通過V4L2介面來實現，不過目前的實現中還是有些例外，引發例外的主要因素是效率問題。  
        由於在USB匯流排上傳輸圖像，受到匯流排速度的限制，特別是在非USB2.0介面的晶片中，所以勢必要採用JPEG或MPEG編碼對資料進行壓縮。V4L2子系統的框架中，包含了對編碼器的支援，但是，也許是筆者對V4L2子系統的學習還不夠深入，其對編碼器的支援應該是對硬體編碼晶片的支援，即使如此，也沒有見到相關的程式碼，而且，在實現中使用的手機主機板上也沒有硬體的編解碼晶片，所以在實現中Camera驅動通過V4L2子系統對應用層提供的是原始的未經編碼的資料流，由應用程式呼叫IJG （ Independent JPEG Group ）JPEG庫函數來實現jpeg的編解碼工作。  
        所以如果通過V4L2的read介面來獲取資料，勢必只能得到原始的圖像資料，而且在V4L2的實現中，通過Read方式獲取的資料，需要通過多次記憶體複製才能到達呼叫者處。所以也會很大程度的影響圖像處理的速度。  
要對圖像進行壓縮，要不在使用者空間呼叫IJG庫，要不在核心中再實現一個JPEG壓縮演算法。按前者來實現的話，涉及到Video Class如何去啟動一個使用者程序來讀取Camera資料並在壓縮後再傳送給核心，也不是完全沒法實現，但是無疑是一個非常糟糕的實現辦法。  
        後者的話，涉及到這個JPEG壓縮演算法應該在什麼地方實現，以及由誰來呼叫的問題（Video Class 還是 V4L2）。考慮到在存在硬體編碼晶片的情況下，該晶片的管理和使用應該會納入V4L2子系統中，所以考慮到相容性，目前實現的方式是將JPEG壓縮演算法作為一個獨立的模組插入核心，由V4L2子系統呼叫相關函數對圖像資料進行壓縮，然後再在Camera驅動中Export一個額外的函數介面，USB Video Class通過該函數介面啟動Camera圖像的讀取和壓縮過程，以區別標準的V4L2子系統的資料處理流程。壓縮後的圖像資料直接寫入通過指針參數傳遞進來的記憶體地址空間中，儘可能的減少記憶體複製的次數，加速圖像的傳遞。  
這樣做帶來的問題就是需要在V4L2架構中新增額外的介面，增加了USB Video Class和V4L2子系統之間的耦合性，不能完全將這兩個模組隔離開來。應該還有更好的解決方案。

### 3.3 JPEG編碼相關

        Jpeg的編解碼，在Linux作業系統中，基本上採用的都是IJG（[www.ijg.org](http://www.ijg.org/)）的JPEG庫函數Libjpeg，這是一個相當可靠穩定和高效的開放原始碼專案，支援JPEG標準（不包括JPEG2000）的絕大多數編碼方式。如非無奈，確實沒有必要另外再寫一個編碼程序。但是由於需要在核心中使用，所以只好自己再編一個了。  
        JPEG編碼相關的程式碼除了IJG的原始碼以外，在網上還可以搜尋到若干，但是無疑IJG的程式碼是最完善的。其它我能搜到的程式碼，多多少少都有一些BUG，而且也只是實現了JPEG標準的最基本的功能（當然，對於Video Class的應用來說已經是足夠了）。最重要的是，多數是用浮點數運算來實現的，撇開速度不說，在本文的實現中OMAP平台的CPU也不支援浮點數運算。。。所以，本文實現中，最終是參考了網上搜到的某個演算法的流程（主要是IJG的架構太複雜，一來沒有時間精力和能力進行完整的分析，二來也不適合在核心中使用如此複雜的架構），在快速離散餘弦變化的整數演算法上仿照了IJG庫的演算法兩者綜合起來完成的。最終的程式碼還有很多需要改進的地方，不過，對於VideoClass來說，應該勉強夠用了。這其中的具體問題，打算在另外單獨的文件中再說明。

### 3.4 作業系統相關

        說作業系統相關，可能說大了一些，這裡主要涉及的內容是在本文的實現中，在WIN2000和WINXP平台的MSN測試中，遇到的一些問題。  
        由於VideoClass的協議只是規定了資料傳輸的格式和內容，對具體實現中的一些細節並沒有作硬性的規定，所以導致有些細節可能存在不相容的實現方式。（當然，我想主要還是本文的實現，由於能力有限，沒有充分考慮到各種情況下的容錯性，如果驅動做得好應該可以避免出現問題）。所以在WIN2000和WINXP的MSN測試中，遇到了一些平台相關的問題，有些功能在2000下能正常工作在XP下存在Bug，有些卻相反。有些已經解決，有些只是猜測了可能的原因，羅列如下：

#### 3.4.1 視訊窗口關閉再打開後，沒有圖像

        開始是在XP的MSN上發現有這樣的問題，2000下沒有，分析BUS資料可以看到，XP在關閉視訊窗口的時候，會執行一個Abort Pipe的操作，這個操作應該會中斷BULK傳輸，但是在裝置端，Gadget底層驅動接收不到這個事件(也有可能是Gadget底層驅動的BUG)，所以在VideoClass中無從得知這個傳輸已經被取消了，這樣睡眠在等待資料傳送完畢或失敗上的執行緒也就無法被喚醒，自然也就不會繼續傳送資料。造成主機端再度打開視訊窗口時接收不到圖像資料。而在2000下的MSN中，關閉視訊窗口的動作系統不會傳送這個Abort Pipe事件，所以也就沒有問題。  
考慮到每次打開視訊窗口的時候，主機端都會設定Streaming Interface的圖像解析度，位元率等參數。而這之後主機端才會讀取圖像資料，所以後來解決的辦法是在主機端設定Streaming Interface的時候，將之前已經放入BULK IN傳輸節點的資料 Dequeue出來，這樣會造成這個傳輸的失敗，從而喚醒睡眠的執行緒。但是如果僅僅這樣做，XP能夠正常工作了，2000又顯示不了圖像了。分析認為由於部分資料丟失，所以造成第一幀圖像的資料是不完整的，無法正常解壓縮顯示，但是XP下的MSN有較好的容錯性，能夠丟棄這一幀圖像，繼續讀取之後的資料，而2000下的MSN容錯能力較差，無法再正常解讀後面的圖像資料。所以最終的解決辦法是在發現傳輸失敗後，將當前這一幀的圖像資料從頭開始重新傳送，這樣在XP和2000下就都能正常工作了。  
不知道這種解決方案是否僅僅是一種治標的方案，有待以後繼續研究。

#### 3.4.2 某些解析度下圖像無法正常顯示

        在Win2000中如果提供160\*120解析度的圖像，圖像非常容易就停止刷新了，而BUS上實際資料還是在傳送的。而在160\*112（兩者都是16的整倍數）的解析度的情況下，就幾乎不會發生這種情況。如果說這有可能還是JPEG的壓縮演算法有點問題，那另外一種情況就一定是XP 和 2000的區別了：如果裝置這端通過描述符和Streaming Interface申明只能支援160\*120 或者 160\*112 的解析度，2000可以接受這種解析度，而XP根本就不能接受，匯流排上的控制傳輸就停止了，在介面上則顯示檢測不到Camera或Camera正在被其它裝置打開佔用，只有在進一步提供更高的320\*240的解析度的情況下，XP才會承認Camera的存在！其它問題倒不大，就是在本文的實現平台上，受軟體編碼JPEG速度的限制，320\*240的解析度下，視訊的幀頻會低一些，影響圖像的流暢性。

### 3.5 其它

#### 3.5.1 特殊效果的控制

        應該說，VideoClass的Control Request基本上涵蓋了V4L2標準介面提供的大部分控制參數，但是，還是有一部分沒有涵蓋，至於特定驅動專有的控制就更無法體現了，尤其是在MSN等應用程式的介面上，更不可能提供這些參數的控制了。但是，我們還是可以想辦法trick過這個問題。  
        比如手機上常見的圖像效果的設定，雖然不是特別有意義，但是既然是很常見的，為什麼不能把它也做到Web Cam中呢？所以，如果一定要做，我們可以利用MSN控制介面上的原有的控制介面，借用其中一兩個控制參數來實現圖像效果的設定。  
        本文的實現中選擇採用色調來控製圖像效果，因為實際上這個參數是很不常用的，甚至只能在XP的高級設定中找到，對於99.9%的使用者我相信都不會去改變這個參數。而它的字面含義與我們實現的功能也不算一點關係都沒有，畢竟有很多效果實際上就是改變一下圖像的顏色（當然還有一部分例外了）。  
        類似的可以用一些我們認為常用的設定替換既有的參數。這樣做的缺點就是控制參數的字面含義與實際功能不太吻合，優點當然就是可以提供給使用者更多更常用的圖像設定。比如設定一個黑白，素描之類的圖像的效果，玩玩抽象派視訊聊天。

[](https://sites.google.com/site/rgbbones2/usbvideo-5.jpg)

圖5  圖像效果設定