**YUV Formats**

**YUV** formats fall into two distinct groups, the [packed formats](http://www.fourcc.org/yuv.php#Packed YUV Formats) where Y, U (Cb) and V (Cr) samples are packed together into macropixels which are stored in a single array, and the [planar formats](http://www.fourcc.org/yuv.php#Planar YUV Formats) where each component is stored as a separate array, the final image being a fusing of the three separate planes.

In the diagrams below, the numerical suffix attached to each Y, U or V sample indicates the sampling position across the image line, so, for example, V0 indicates the leftmost V sample and Yn indicates the Y sample at the (n+1)th pixel from the left.

Subsampling intervals in the horizontal and vertical directions may merit some explanation. The horizontal subsampling interval describes how frequently across a line a sample of that component is taken while the vertical interval describes on which lines samples are taken. For example, UYVY format has a horizontal subsampling period of 2 for both the U and V components indicating that U and V samples are taken for every second pixel across a line. Their vertical subsampling period is 1 indicating that U and V samples are taken on each line of the image.

For YVU9, though, the vertical subsampling interval is 4. This indicates that U and V samples are only taken on every fourth line of the original image. Since the horizontal sampling period is also 4, a single U and a single V sample are taken for each square block of 16 image pixels.

Also, if you are interested in YCrCb to RGB conversion, you may find [this page](http://www.fourcc.org/fccyvrgb.php) helpful.

People reading this page may be interested in a freeware codec from Drastic Technologies which allegedly handles the vast majority of **YUV** formats listed here. I've not tried it but you can find it [here](http://www.drastictech.com/download_codec.html).

**Packed YUV Formats**

|  |  |  |  |
| --- | --- | --- | --- |
| **Label** | **FOURCC in Hex** | **Bits per pixel** | **Description** |
| [AYUV](http://www.fourcc.org/yuv.php#AUV) | 0x56555941 | 32 | Combined **YUV** and alpha |
| [CLJR](http://www.fourcc.org/yuv.php#CLJR) | 0x524A4C43 | 8 | Cirrus Logic format with 4 pixels packed into a u\_int32. A form of **YUV** 4:1:1 wiht less than 8 bits per Y, U and V sample. |
| [cyuv](http://www.fourcc.org/yuv.php#cyuv) | 0x76757963 | 16 | Essentially a copy of UYVY except that the sense of the height is reversed - the image is upside down with respect to the UYVY version. |
| [GREY](http://www.fourcc.org/yuv.php#GREY) | 0x59455247 | 8 | Apparently a duplicate of Y800 (and also, presumably, "Y8 ") |
| IRAW | 0x57615349 | ? | Intel uncompressed **YUV**. I have no information on this format - can you help? |
| [IUYV](http://www.fourcc.org/yuv.php#IY41) | 0x56595549 | 16 | Interlaced version of UYVY (line order 0, 2, 4,....,1, 3, 5....) registered by Silviu Brinzei of [LEAD Technologies](http://www.leadtools.com/). |
| [IY41](http://www.fourcc.org/yuv.php#IY41) | 0x31345949 | 12 | Interlaced version of Y41P (line order 0, 2, 4,....,1, 3, 5....) registered by Silviu Brinzei of [LEAD Technologies](http://www.leadtools.com/). |
| [IYU1](http://www.fourcc.org/yuv.php#IYU1) | 0x31555949 | 12 | 12 bit format used in mode 2 of the IEEE 1394 Digital Camera 1.04 spec. This is equivalent to [Y411](http://www.fourcc.org/yuv.php#Y411) |
| [IYU2](http://www.fourcc.org/yuv.php#IYU2) | 0x32555949 | 24 | 24 bit format used in mode 0 of the IEEE 1394 Digital Camera 1.04 spec |
| [HDYC](http://www.fourcc.org/yuv.php#HDYC) | 0x43594448 | 16 | **YUV** 4:2:2 (Y sample at every pixel, U and V sampled at every second pixel horizontally on each line). A macropixel contains 2 pixels in 1 u\_int32. This is a suplicate of [UYVY](http://www.fourcc.org/yuv.php#UYVY) except that the color components use the BT709 color space (as used in HD video). |
| [UYNV](http://www.fourcc.org/yuv.php#UYVY) | 0x564E5955 | 16 | A direct copy of [UYVY](http://www.fourcc.org/yuv.php#UYVY) registered by NVidia to work around problems in some old codecs which did not like hardware which offered more than 2 UYVY surfaces. |
| UYVP | 0x50565955 | 24? | YCbCr 4:2:2 extended precision 10-bits per component in U0Y0V0Y1 order. Registered by [Rich Ehlers](mailto:ehlers@es.com) of Evans & Sutherland. (Awaiting confirmation of component packing structure) |
| [UYVY](http://www.fourcc.org/yuv.php#UYVY) | 0x59565955 | 16 | **YUV** 4:2:2 (Y sample at every pixel, U and V sampled at every second pixel horizontally on each line). A macropixel contains 2 pixels in 1 u\_int32. |
| [V210](http://www.fourcc.org/yuv.php#V210) | 0x30313256 | 32 | 10-bit 4:2:2 YCrCb equivalent to the Quicktime format of the same name. |
| V422 | 0x32323456 | 16 | I am told that this is an upside down version of UYVY. |
| V655 | 0x35353656 | 16? | 16 bit **YUV** 4:2:2 format registered by Vitec Multimedia. I have no information on the component ordering or packing. |
| VYUY | 0x59555956 | ? | ATI Packed **YUV** Data (format unknown but you can get hold of a codec supporting it [here](http://www.ati.com/support/drivers/misc/ATIVCRX.html)) |
| [Y422](http://www.fourcc.org/yuv.php#UYVY) | 0x32323459 | 16 | Direct copy of UYVY as used by ADS Technologies Pyro WebCam firewire camera. |
| [YUY2](http://www.fourcc.org/yuv.php#YUY2) | 0x32595559 | 16 | **YUV** 4:2:2 as for UYVY but with different component ordering within the u\_int32 macropixel. |
| [YUYV](http://www.fourcc.org/yuv.php#YUYV) | 0x56595559 | 16 | Duplicate of YUY2 |
| [YUNV](http://www.fourcc.org/yuv.php#YUY2) | 0x564E5559 | 16 | A direct copy of [YUY2](http://www.fourcc.org/yuv.php#YUY2) registered by NVidia to work around problems in some old codecs which did not like hardware which offered more than 2 YUY2 surfaces. |
| [YVYU](http://www.fourcc.org/yuv.php#YVYU) | 0x55595659 | 16 | **YUV** 4:2:2 as for UYVY but with different component ordering within the u\_int32 macropixel. |
| [Y41P](http://www.fourcc.org/yuv.php#Y41P) | 0x50313459 | 12 | **YUV** 4:1:1 (Y sample at every pixel, U and V sampled at every fourth pixel horizontally on each line). A macropixel contains 8 pixels in 3 u\_int32s. |
| [Y411](http://www.fourcc.org/yuv.php#Y411) | 0x31313459 | 12 | **YUV** 4:1:1 with a packed, 6 byte/4 pixel macroblock structure. |
| [Y211](http://www.fourcc.org/yuv.php#Y211) | 0x31313259 | 8 | Packed **YUV** format with Y sampled at every second pixel across each line and U and V sampled at every fourth pixel. |
| [Y41T](http://www.fourcc.org/yuv.php#Y41T) | 0x54313459 | 12 | Format as for Y41P but the lsb of each Y component is used to signal pixel transparency . |
| [Y42T](http://www.fourcc.org/yuv.php#Y42T) | 0x54323459 | 16 | Format as for UYVY but the lsb of each Y component is used to signal pixel transparency . |
| [YUVP](http://www.fourcc.org/yuv.php#YUVP) | 0x50565559 | 24? | YCbCr 4:2:2 extended precision 10-bits per component in Y0U0Y1V0 order. Registered by [Rich Ehlers](mailto:ehlers@es.com) of Evans & Sutherland. |
| [Y800](http://www.fourcc.org/yuv.php#Y800) | 0x30303859 | 8 | Simple, single Y plane for monochrome images. |
| [Y8](http://www.fourcc.org/yuv.php#Y8) | 0x20203859 | 8 | Duplicate of Y800 as far as I can see. |
| [Y16](http://www.fourcc.org/yuv.php#Y16) | 0x20363159 | 16 | 16-bit uncompressed greyscale image. |

**AYUV**

This is a 4:4:4 **YUV** format with 8 bit samples for each component along with an 8 bit alpha blend value per pixel. Component ordering is A Y U V (as the name suggests).

**UYVY (and Y422 and UYNV and** **HDYC)**

UYVY is probably the most popular of the various **YUV** 4:2:2 formats. It is output as the format of choice by the Radius Cinepak codec and is often the second choice of software MPEG codecs after YV12.

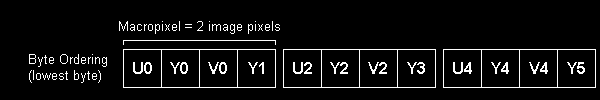
Y422 and UYNV appear to be direct equivalents to the original UYVY.

HDYC is equivalent in layout but pixels are described using the BT709 color space as used in HD video systems rather than the BT470 SD video color space typically used. Apparently there is a description in the DeckLink DirectShow SDK documentation at <http://blackmagic-design.com/support/software/archive/>, find **DeckLink SDK 5.6.2 for Windows XP** and download <http://blackmagic-design.com/support/software/register.asp?softID=108>, set product to None, serial no is not required), see "Video Formats" section.

|  |  |  |
| --- | --- | --- |
|  | **Horizontal** | **Vertical** |
| **Y Sample Period** | 1 | 1 |
| **V Sample Period** | 2 | 1 |
| **U Sample Period** | 2 | 1 |

Effective bits per pixel : 16

Positive biHeight implies top-down imge (top line first)



**IUYV**

IUYV is basically the same as UYVY with the exception that the data is interlaced. Lines are ordered 0,2,4,....,1,3,5.... instead of 0,1,2,3,4,5,....

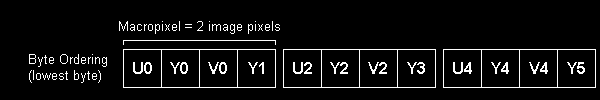
**cyuv**

This FOURCC, allegedly registered by Creative Labs, is essentially a duplicate of UYVY. The only difference is that the image is flipped vertically, the first u\_int16 in the buffer representing the bottom line of the viewed image. Note that the FOURCC is comprised of lower case characters (so much for the upper case convention !)

|  |  |  |
| --- | --- | --- |
|  | **Horizontal** | **Vertical** |
| **Y Sample Period** | 1 | 1 |
| **V Sample Period** | 2 | 1 |
| **U Sample Period** | 2 | 1 |

Effective bits per pixel : 16

Positive biHeight implies bottom-up image (botton line first)



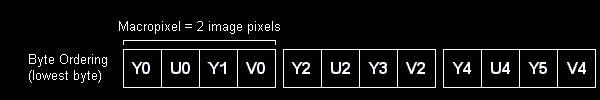
**YUY2 (and** **YUNV and** **V422 and** **YUYV)**

YUY2 is another in the family of **YUV** 4:2:2 formats and appears to be used by all the same codecs as UYVY.

|  |  |  |
| --- | --- | --- |
|  | **Horizontal** | **Vertical** |
| **Y Sample Period** | 1 | 1 |
| **V Sample Period** | 2 | 1 |
| **U Sample Period** | 2 | 1 |

Effective bits per pixel : 16

Positive biHeight implies top-down image (top line first)



There is a [help page here](http://www.fourcc.org/fccyuy2.php) which contains information on playing AVIs which include video stored in YUY2 format.

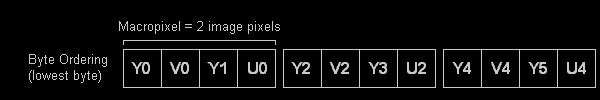
**YVYU**

Despite being a simple byte ordering change from YUY2 or UYVY, YVYU seems to be seen somewhat less often than the other two formats defined above.

|  |  |  |
| --- | --- | --- |
|  | **Horizontal** | **Vertical** |
| **Y Sample Period** | 1 | 1 |
| **V Sample Period** | 2 | 1 |
| **U Sample Period** | 2 | 1 |

Effective bits per pixel : 16

Positive biHeight implies top-down image (top line first)



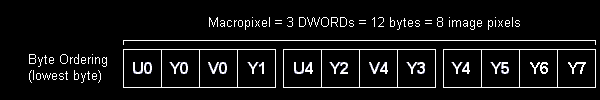
**Y41P**

This **YUV** 4:1:1 format is registered as a PCI standard format. Mediamatics MPEG 1 engine is the only codec (other than a Brooktree internal one) that I know of that can generate it.

|  |  |  |
| --- | --- | --- |
|  | **Horizontal** | **Vertical** |
| **Y Sample Period** | 1 | 1 |
| **V Sample Period** | 4 | 1 |
| **U Sample Period** | 4 | 1 |

Effective bits per pixel : 12

Positive biHeight implies top-down image (top line first)



**Y411**

I was originally told that this was a duplicate of [Y41P](http://www.fourcc.org/yuv.php#Y41P) however it seems that this is not the case after all. Y411 is a packed **YUV** 4:1:1 format with a 6 pixel macroblock structure containing 4 pixels. Component packing order is:

**U2 Y0 Y1 V2 Y2 Y3**

I have not been able to find 100% confirmation of the position for the U and V samples. I suspect that the chroma samples are probably both taken at the position of Y2 but this is a guess just now.

I have recently been informed that this format is identical to [IYU1](http://www.fourcc.org/yuv.php#IYU1).

|  |  |  |
| --- | --- | --- |
|  | **Horizontal** | **Vertical** |
| **Y Sample Period** | 1 | 1 |
| **V Sample Period** | 4 | 1 |
| **U Sample Period** | 4 | 1 |

Effective bits per pixel : 12

Positive biHeight implies top-down image (top line first)

**IY41**

IY41 is basically the same as Y41P with the exception that the data is interlaced. Lines are ordered 0,2,4,....,1,3,5.... instead of 0,1,2,3,4,5,....

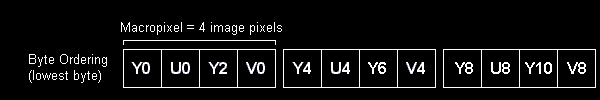
**Y211**

I have yet to find anything that will output Y211 ! The format looks very much like the missing **YUV** 4:2:2 ordering but Y samples are only taken on every second pixel. Think of it as a half width 4:2:2 image and double the width on display.

|  |  |  |
| --- | --- | --- |
|  | **Horizontal** | **Vertical** |
| **Y Sample Period** | 2 | 1 |
| **V Sample Period** | 4 | 1 |
| **U Sample Period** | 4 | 1 |

Effective bits per pixel : 8

Positive biHeight implies top-down image (top line first)



**Y41T**

This format is identical to [Y41P](http://www.fourcc.org/yuv.php#Y41P) except for the fact that the least significant bit of each Y component forms a chromakey channel. If this bit is set, the **YUV** image pixel is displayed, if cleared, the pixel is transparent (and the underlying graphics pixel is shown).

Positive biHeight implies top-down image (top line first)

**Y42T**

This format is identical to [UYVY](http://www.fourcc.org/yuv.php#UYVY) except for the fact that the least significant bit of each Y component forms a chromakey channel. If this bit is set, the **YUV** image pixel is displayed, if cleared, the pixel is transparent (and the underlying graphics pixel is shown).

Positive biHeight implies top-down image (top line first)

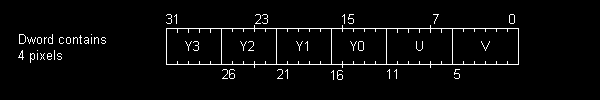
**CLJR**

Cirrus Logic's format packs 4 pixel samples into a single u\_int32 by sacrificing precision on each sample. Y samples are truncated to 5 bits each, U and V have 6 bits per sample.

|  |  |  |
| --- | --- | --- |
|  | **Horizontal** | **Vertical** |
| **Y Sample Period** | 1 | 1 |
| **V Sample Period** | 4 | 1 |
| **U Sample Period** | 4 | 1 |

Effective bits per pixel : 8

Positive biHeight implies top-down image (top line first)



**IYU1**

The IYU1 format is a 12 bit format used in mode 2 of the IEEE 1394 Digital Camera 1.04 spec ("1394-based Digital Camera Specification, Version 1.04, August 9, 1996", page 14.). The format, a duplicate of [Y411](http://www.fourcc.org/yuv.php#Y411), is **YUV** (4:1:1) according to the following pattern:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Byte** | **0** | **1** | **2** | **3** | **4** | **5** |
| **Sample** | U(K+0) | Y(K+0) | Y(K+1) | V(K+0) | Y(K+2) | Y(K+3) |

|  |  |  |
| --- | --- | --- |
|  | **Horizontal** | **Vertical** |
| **Y Sample Period** | 1 | 1 |
| **V Sample Period** | 4 | 1 |
| **U Sample Period** | 4 | 1 |

**IYU2**

The IYU2 format is a 24 bit format used in mode 0 of the IEEE 1394 Digital Camera 1.04 spec (ibid.) The format is **YUV** (4:4:4) according to the following pattern:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Byte** | **0** | **1** | **2** | **3** | **4** | **5** |
| **Sample** | U(K+0) | Y(K+0) | V(K+0) | U(K+1) | Y(K+1) | V(K+1) |

|  |  |  |
| --- | --- | --- |
|  | **Horizontal** | **Vertical** |
| **Y Sample Period** | 1 | 1 |
| **V Sample Period** | 1 | 1 |
| **U Sample Period** | 1 | 1 |

**YUVP**

This is another format similar to YUY2 and it's aliases. The difference here is that each Y, U and V samples is 10 bits rather than 8. I am still waiting to hear how the samples are packed - is a macropixel just 5 bytes long with all the samples packed together or is there more to it than this?

**V210**

[AJA Video Systems](http://www.aja.com/) have implemented this Quicktime format for Windows. It is a 10 bit per component, YCrCb 4:2:2 format in which samples for 5 pixels are packed into 4 4-byte little endian words. Rather than repeat the details here, I suggest looking at the [original definition on the Quicktime web site](http://developer.apple.com/quicktime/icefloe/dispatch019.html#v210).

Supposedly there are images described as "**YUV**10" that are formatted similarly to this aside from the bit ordering (the correspondent mentioned having to run ntoh on the pixel data to reformat from **YUV**10 to V210. Despite 20 years of C, I've not heard of ntoh but I suspect it performs big-endian to little-endian conversion).

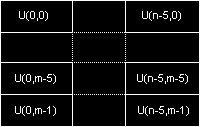
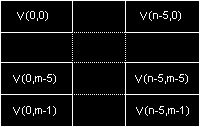
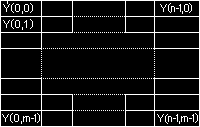
**Planar YUV Formats**

|  |  |  |  |
| --- | --- | --- | --- |
| **Label** | **FOURCC in Hex** | **Bits per pixel** | **Description** |
| [YVU9](http://www.fourcc.org/yuv.php#YVU9) | 0x39555659 | 9 | 8 bit Y plane followed by 8 bit 4x4 subsampled V and U planes. Registered by Intel. |
| [**YUV**9](http://www.fourcc.org/yuv.php#YUV9) | 0x39565559 | 9? | Registered by Intel., this is the format used internally by Indeo video code |
| [IF09](http://www.fourcc.org/yuv.php#IF09) | 0x39304649 | 9.5 | As YVU9 but an additional 4x4 subsampled plane is appended containing delta information relative to the last frame. (Bpp is reported as 9) |
| [YV16](http://www.fourcc.org/yuv.php#YV16) | 0x36315659 | 16 | 8 bit Y plane followed by 8 bit 2x1 subsampled V and U planes. |
| [YV12](http://www.fourcc.org/yuv.php#YV12) | 0x32315659 | 12 | 8 bit Y plane followed by 8 bit 2x2 subsampled V and U planes. |
| [I420](http://www.fourcc.org/yuv.php#IYUV) | 0x30323449 | 12 | 8 bit Y plane followed by 8 bit 2x2 subsampled U and V planes. |
| [IYUV](http://www.fourcc.org/yuv.php#IYUV) | 0x56555949 | 12 | Duplicate FOURCC, identical to I420. |
| [NV12](http://www.fourcc.org/yuv.php#NV12) | 0x3231564E | 12 | 8-bit Y plane followed by an interleaved U/V plane with 2x2 subsampling |
| **[NV21](http://www.fourcc.org/yuv.php" \l "NV21)** | 0x3132564E | 12 | As NV12 with U and V reversed in the interleaved plane |
| [IMC1](http://www.fourcc.org/yuv.php#IMC1) | 0x31434D49 | 12 | As YV12 except the U and V planes each have the same stride as the Y plane |
| [IMC2](http://www.fourcc.org/yuv.php#IMC2) | 0x32434D49 | 12 | Similar to IMC1 except that the U and V lines are interleaved at half stride boundaries |
| [IMC3](http://www.fourcc.org/yuv.php#IMC3) | 0x33434D49 | 12 | As IMC1 except that U and V are swapped |
| [IMC4](http://www.fourcc.org/yuv.php#IMC4) | 0x34434D49 | 12 | As IMC2 except that U and V are swapped |
| [CLPL](http://www.fourcc.org/yuv.php#CLPL) | 0x4C504C43 | 12 | Format similar to YV12 but including a level of indirection. |
| Y41B | 0x42313459 | 12? | Weitek format listed as "**YUV** 4:1:1 planar". I have no other information on this format. |
| Y42B | 0x42323459 | 16? | Weitek format listed as "**YUV** 4:2:2 planar". I have no other information on this format. |
| [Y800](http://www.fourcc.org/yuv.php#Y800) | 0x30303859 | 8 | Simple, single Y plane for monochrome images. |
| [Y8](http://www.fourcc.org/yuv.php#Y800) | 0x20203859 | 8 | Duplicate of Y800 as far as I can see. |
| [CXY1](http://www.fourcc.org/yuv.php#CXY1) | 0x31595843 | 12 | Awaiting clarification of format. |
| [CXY2](http://www.fourcc.org/yuv.php#CXY2) | 0x32595842 | 16 | Awaiting clarification of format. |

**YVU9**

This format dates back to the days of the ActionMedia II adapter and comprises an NxN plane of Y samples, 8 bits each, followed by (N/4)x(N/4) V and U planes.

|  |  |  |
| --- | --- | --- |
|  | **Horizontal** | **Vertical** |
| **Y Sample Period** | 1 | 1 |
| **V Sample Period** | 4 | 4 |
| **U Sample Period** | 4 | 4 |



Positive biHeight implies top-down image (top line first)

ATI has a codec supporting this format that you can download from [here](http://www.ati.com/support/drivers/misc/ATIVCRX.html).

**YUV9**

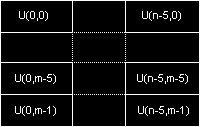
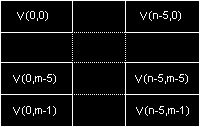
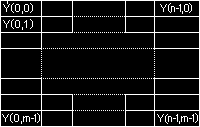
[Intel's web site](http://www.intel.com/support/createshare/sb/cs-011802.htm) states that **YUV**9 is "the color encoding scheme used in Indeo video technology. The **YUV**9 format stores information in 4x4 pixel blocks. Sixteen bytes of luminance are stored for every 1 byte of chrominance. For example, a 640x480 image will have 307,200 bytes of luminance and 19,200 bytes of chrominance." This sounds exactly the same as [YVU9](http://www.fourcc.org/yuv.php#YVU9) to me. Anyone know if there is any difference?

**IF09**

A derivative of YVU9, IF09 contains the basic 3 planes for Y, V and U followed by an additional (N/4)x(N/4) plane of "skip blocks". This final plane forms a basic delta encoding scheme which can be used by a displayer to decide which pixels in the image are unchanged from the previous displayed frame. The strange number of bits per pixel listed for the format results from the fact that an NxN image is described using N2+3(N/4)2 bytes.

This format is generated by Intel's Indeo codecs though users should beware - the original 32 bit Indeo 3.2 shipped with Windows 95 and the beta levels of Indeo 4.1 contain bugs which cause them to generate protection faults when using IF09. Fixed versions of these codecs are available from Intel.

|  |  |  |
| --- | --- | --- |
|  | **Horizontal** | **Vertical** |
| **Y Sample Period** | 1 | 1 |
| **V Sample Period** | 4 | 4 |
| **U Sample Period** | 4 | 4 |



Positive biHeight implies top-down image (top line first)

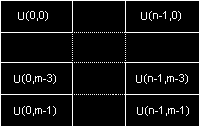
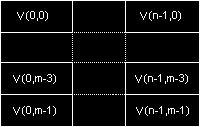
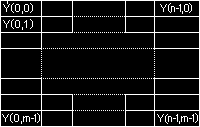
**Delta plane definition**

To be completed...

**YV12**

This is the format of choice for many software MPEG codecs. It comprises an NxM Y plane followed by (N/2)x(M/2) V and U planes.

|  |  |  |
| --- | --- | --- |
|  | **Horizontal** | **Vertical** |
| **Y Sample Period** | 1 | 1 |
| **V Sample Period** | 2 | 2 |
| **U Sample Period** | 2 | 2 |



Positive biHeight implies top-down image (top line first)

ATI says they have [a download which contains this codec](https://support.ati.com/ics/support/KBAnswer.asp?questionID=36&fieldOffset=17&hitOffset=908+907+889+887+881+880+12+6+4+) but I can't find it on their site. If you would like something similar for Quicktime, [try here](http://www.unthinkable.com/yv12codec.htm).

**YV16**

This format is basically a version of [YV12](http://www.fourcc.org/yuv.php#YV12) with higher chroma resolution. It comprises an NxM Y plane followed by (N/2)xM U and V planes.

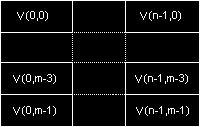
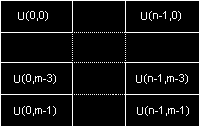
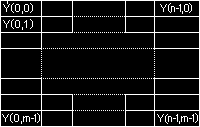
|  |  |  |
| --- | --- | --- |
|  | **Horizontal** | **Vertical** |
| **Y Sample Period** | 1 | 1 |
| **V Sample Period** | 2 | 1 |
| **U Sample Period** | 2 | 1 |

**IYUV and I420**

These formats are identical to YV12 except that the U and V plane order is reversed. They comprise an NxN Y plane followed by (N/2)x(N/2) U and V planes. Full marks to Intel for registering the same format twice and full marks to Microsoft for not picking up on this and rejecting the second registration.

*(Note: There is some confusion over these formats thanks to the definitions on* [*Microsoft's site*](http://msdn.microsoft.com/library/en-us/wmform95/htm/fourccconstants.asp) *which tend to suggest that the two FOURCCs are different. One is described as a 4:2:0 format while the other is described as 4:1:1. Later, however, the same page states that YV12 is the same as both of these with the U and V plane order reversed. I would consider 4:2:0 to imply 1 chroma sample for every 2x2 luma block and 4:1:1 to imply 1 chroma sample for every 4x1 luma block but it seems as if the Microsoft writer may have been using the terms interchangeably. If you know these formats, please could you* [*let me know*](http://www.fourcc.org/email.php) *whether the definition here is correct or whether I need to update one or other?)*

|  |  |  |
| --- | --- | --- |
|  | **Horizontal** | **Vertical** |
| **Y Sample Period** | 1 | 1 |
| **V Sample Period** | 2 | 2 |
| **U Sample Period** | 2 | 2 |

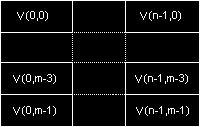
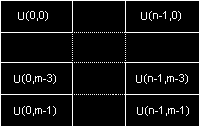
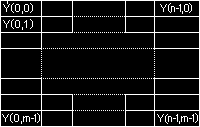


Positive biHeight implies top-down image (top line first)

**CLPL**

This format introduces an extra level of indirection in the process of accessing **YUV** pixels in the surface. Locking the DirectDraw or DCI CLPL surface returns a pointer which itself points to three other pointers. These pointers respectively point to an NxN Y plane, an (N/2)x(N/2) U plane and an (N/2)x(N/2) V plane. The Y plane pointer retrieved is (allegedly) valid even when the surface is subsequently unlocked but the U and V pointers can only be used with a lock held (as you should be doing anyway if adhereing to the DirectDraw/DCI spec).

|  |  |  |
| --- | --- | --- |
|  | **Horizontal** | **Vertical** |
| **Y Sample Period** | 1 | 1 |
| **V Sample Period** | 2 | 2 |
| **U Sample Period** | 2 | 2 |

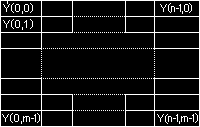


Positive biHeight implies top-down image (top line first)

**Y800**

This format contains only a single, 8 bit Y plane for monochrome images. Apparent duplicate FOURCCs are "Y8" and "GREY".

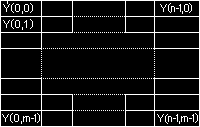
|  |  |  |
| --- | --- | --- |
|  | **Horizontal** | **Vertical** |
| **Y Sample Period** | 1 | 1 |
| **V Sample Period** | N/A | N/A |
| **U Sample Period** | N/A | N/A |



**Y16**

This format contains only a single, 16 bit Y plane for monochrome images. Each pixel is represented by a 16 bit, little endian luminance sample.

|  |  |  |
| --- | --- | --- |
|  | **Horizontal** | **Vertical** |
| **Y Sample Period** | 1 | 1 |
| **V Sample Period** | N/A | N/A |
| **U Sample Period** | N/A | N/A |



**NV12**

**YUV** 4:2:0 image with a plane of 8 bit Y samples followed by an interleaved U/V plane containing 8 bit 2x2 subsampled colour difference samples.

|  |  |  |
| --- | --- | --- |
|  | **Horizontal** | **Vertical** |
| **Y Sample Period** | 1 | 1 |
| **V (Cr) Sample Period** | 2 | 2 |
| **U (Cb) Sample Period** | 2 | 2 |

Microsoft defines this format as follows:

"A format in which all Y samples are found first in memory as an array of unsigned char with an even number of lines (possibly with a larger stride for memory alignment), followed immediately by an array of unsigned char containing interleaved Cb and Cr samples (such that if addressed as a little-endian WORD type, Cb would be in the LSBs and Cr would be in the MSBs) with the same total stride as the Y samples. This is the preferred 4:2:0 pixel format."

**NV21**

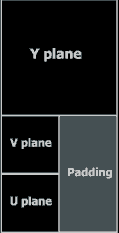
**YUV** 4:2:0 image with a plane of 8 bit Y samples followed by an interleaved V/U plane containing 8 bit 2x2 subsampled chroma samples. The same as [NV12](http://www.fourcc.org/yuv.php#NV12) except the interleave order of U and V is reversed.

|  |  |  |
| --- | --- | --- |
|  | **Horizontal** | **Vertical** |
| **Y Sample Period** | 1 | 1 |
| **V (Cr) Sample Period** | 2 | 2 |
| **U (Cb) Sample Period** | 2 | 2 |

Microsoft defines this format as follows:

"The same as [NV12](http://www.fourcc.org/yuv.php#NV12), except that Cb and Cr samples are swapped so that the chroma array of unsigned char would have Cr followed by Cb for each sample (such that if addressed as a little-endian WORD type, Cr would be in the LSBs and Cb would be in the MSBs)."

**IMC1**

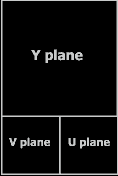
Similar to [YV12](http://www.fourcc.org/yuv.php#YV12), this format comprises an NxN Y plane followed by (N/2)x(N/2) U and V planes. The U and V planes have the same stride as the Y plane and are restricted to start on 16 line boundaries.

|  |  |  |
| --- | --- | --- |
|  | **Horizontal** | **Vertical** |
| **Y Sample Period** | 1 | 1 |
| **V (Cr) Sample Period** | 2 | 2 |
| **U (Cb) Sample Period** | 2 | 2 |

Microsoft defines this format as follows:

"The same as [YV12](http://www.fourcc.org/yuv.php#YV12), except that the stride of the Cb and Cr planes is the same as the stride in the Y plane. The Cb and Cr planes are also restricted to fall on memory boundaries that are a multiple of 16 lines (a restriction that has no effect on usage for the standard formats, since the standards all use 16×16 macroblocks)."

**IMC2**

Similar to [YV12](http://www.fourcc.org/yuv.php#YV12), this format comprises an NxN Y plane followed by "rectangularly adjacent" (N/2)x(N/2) U and V planes. Lines of U and V pixels are interleaved at half stride boundaries below the Y plane.

|  |  |  |
| --- | --- | --- |
|  | **Horizontal** | **Vertical** |
| **Y Sample Period** | 1 | 1 |
| **V (Cr) Sample Period** | 2 | 2 |
| **U (Cb) Sample Period** | 2 | 2 |

Microsoft defines this format as follows:

"The same as [IMC1](http://www.fourcc.org/yuv.php#IMC1), except that Cb and Cr lines are interleaved at half-stride boundaries. In other words, each full-stride line in the chrominance area starts with a line of Cr, followed by a line of Cb that starts at the next half-stride boundary. (This is a more address-space-efficient format than [IMC1](http://www.fourcc.org/yuv.php#IMC1), cutting the chrominance address space in half, and thus cutting the total address space by 25%.) This runs a close second in preference relative to [NV12](http://www.fourcc.org/yuv.php#NV12), but [NV12](http://www.fourcc.org/yuv.php#NV12) appears to be more popular."

**IMC3**

The same as [IMC1](http://www.fourcc.org/yuv.php#IMC1) except for swapping the U and V order.

**IMC4**

The same as [IMC2](http://www.fourcc.org/yuv.php#IMC2) except for swapping the U and V order.

**CXY1**

Planar **YUV** 4:1:1 format registered by Conexant. Awaiting clarification of pixel component ordering.

**CXY2**

Planar **YUV** 4:2:2 format registered by Conexant. Awaiting clarification of pixel component ordering.

**[分析YUV数据](http://blog.csdn.net/zhengxu25689/article/details/6460614)**

做视频采集与处理，自然少不了要学会分析**YUV**数据。因为从采集的角度来说，一般的视频采集芯片输出的码流一般都是**YUV**数据流的形式，而从视频处理（例如H.264、MPEG视频编解码）的角度来说，也是在原始**YUV**码流进行编码和解析，所以，了解如何分析**YUV**数据流对于做视频领域的人而言，至关重要。本文就是根据我的学习和了解，简单地介绍如何分析**YUV**数据流。

**YUV**，分为三个分量，“Y”表示明亮度（Luminance或Luma），也就是灰度值；而“U”和“V” 表示的则是色度（Chrominance或Chroma），作用是描述影像色彩及饱和度，用于指定像素的颜色。

与我们熟知的RGB类似，**YUV**也是一种颜色编码方法，主要用于电视系统以及模拟视频领域，它将亮度信息（Y）与色彩信息（UV）分离，没有UV信息一样可以显示完整的图像，只不过是黑白的，这样的设计很好地解决了彩色电视机与黑白电视的兼容问题。并且，**YUV**不像RGB那样要求三个独立的视频信号同时传输，所以用**YUV**方式传送占用极少的频宽。

好了，言归正传，谈谈如何分析**YUV**码流吧。**YUV**码流有多种不同的格式，要分析**YUV**码流，就必须搞清楚你面对的到底是哪一种格式，并且必须搞清楚这种格式的**YUV**采样和分布情况。下面我将介绍几种常用的**YUV**码流格式，供大家参考。

**1. 采样方式**

**YUV**码流的存储格式其实与其采样的方式密切相关，主流的采样方式有三种，**YUV**4:4:4，**YUV**4:2:2，**YUV**4:2:0，关于其详细原理，可以通过网上其它文章了解，这里我想强调的是如何根据其采样格式来从码流中还原每个像素点的**YUV**值，因为只有正确地还原了每个像素点的**YUV**值，才能通过**YUV**与RGB的转换公式提取出每个像素点的RGB值，然后显示出来。

用三个图来直观地表示采集的方式吧，以黑点表示采样该像素点的Y分量，以空心圆圈表示采用该像素点的UV分量。

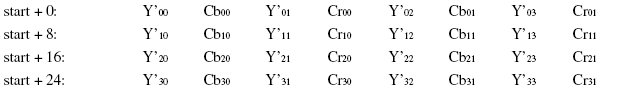
先记住下面这段话，以后提取每个像素的**YUV**分量会用到。

1. **YUV** 4:4:4采样，每一个Y对应一组UV分量。
3. **YUV** 4:2:2采样，每两个Y共用一组UV分量。
4. **YUV** 4:2:0采样，每四个Y共用一组UV分量。

**2. 存储方式**

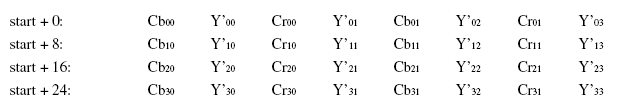
下面我用图的形式给出常见的**YUV**码流的存储方式，并在存储方式后面附有取样每个像素点的**YUV**数据的方法，其中，Cb、Cr的含义等同于U、V。

**（1） YUVY 格式 （属于YUV422）**

[](http://img1.51cto.com/attachment/201104/202410157.png)

YUYV为**YUV**422采样的存储格式中的一种，相邻的两个Y共用其相邻的两个Cb、Cr，分析，对于像素点Y'00、Y'01 而言，其Cb、Cr的值均为 Cb00、Cr00，其他的像素点的**YUV**取值依次类推。

**（2） UYVY 格式 （属于YUV422）**

[](http://img1.51cto.com/attachment/201104/202455202.png)

UYVY格式也是**YUV**422采样的存储格式中的一种，只不过与YUYV不同的是UV的排列顺序不一样而已，还原其每个像素点的**YUV**值的方法与上面一样。

**（3） YUV422P（属于YUV422）**

**YUV**422P也属于**YUV**422的一种，它是一种Plane模式，即打包模式，并不是将**YUV**数据交错存储，而是先存放所有的Y分量，然后存储所有的U（Cb）分量，最后存储所有的V（Cr）分量，如上图所示。其每一个像素点的**YUV**值提取方法也是遵循**YUV**422格式的最基本提取方法，即两个Y共用一个UV。比如，对于像素点Y'00、Y'01 而言，其Cb、Cr的值均为 Cb00、Cr00。

**（4）YV12，YU12格式（属于YUV420）**

YU12和YV12属于**YUV**420格式，也是一种Plane模式，将Y、U、V分量分别打包，依次存储。其每一个像素点的**YUV**数据提取遵循**YUV**420格式的提取方式，即4个Y分量共用一组UV。注意，上图中，Y'00、Y'01、Y'10、Y'11共用Cr00、Cb00，其他依次类推。

**（5）NV12、NV21（属于YUV420）**

NV12和**NV21**属于**YUV**420格式，是一种two-plane模式，即Y和UV分为两个Plane，但是UV（CbCr）为交错存储，而不是分为三个plane。其提取方式与上一种类似，即Y'00、Y'01、Y'10、Y'11共用Cr00、Cb00

**3. 总结**

几种常见的**YUV**码流格式就简单地列在上面了，大家在处理**YUV**码流前，先了解清楚自己的码流到底属于哪一种，然后对应进行处理。

最后，再回答一个疑问，即分析清楚**YUV**码流格式了，我们可以做什么？最常用的一点就是，提取出所有的Y分量，然后利用vc或者matlab把你采集的图像的灰度值（Y分量）显示处理，这样你就可以很快地知道你采集的图像是否有问题了。后面我将继续写一些文章讲述如何提取、转换、显示这些**YUV**原始码流，有兴趣可以继续关注，欢迎留言讨论。

YUV是指亮度参量和色度参量分开表示的像素格式，而这样分开的好处就是不但可以避免相互干扰，还可以降低色度的采样率而不会对图像质量影响太大。YUV是一个比较笼统地说法，针对它的具体排列方式，可以分为很多种具体的格式。

**YUV格式解析1（播放器——project2）**

根据板卡api设计实现yuv420格式的视频播放器

打开\*.mp4;\*.264类型的文件，实现其播放。

使用的视频格式是YUV420格式

YUV格式通常有两大类：打包（packed）格式和平面（planar）格式。前者将YUV分量存放在同一个数组中，通常是几个相邻的像素组成一个宏像素（macro-pixel）；而后者使用三个数组分开存放YUV三个分量，就像是一个三维平面一样。表2.3中的YUY2到Y211都是打包格式，而IF09到YVU9都是平面格式。（注意：在介绍各种具体格式时，YUV各分量都会带有下标，如Y0、U0、V0表示第一个像素的YUV分量，Y1、U1、V1表示第二个像素的YUV分量，以此类推。）

MEDIASUBTYPE\_YUY2 **YUY2**格式，以4:2:2方式打包

MEDIASUBTYPE\_YUYV **YUYV**格式（实际格式与YUY2相同）

MEDIASUBTYPE\_YVYU **YVYU**格式，以4:2:2方式打包

MEDIASUBTYPE\_UYVY **UYVY**格式，以4:2:2方式打包

MEDIASUBTYPE\_AYUV 带Alpha通道的4:4:4 **YUV**格式

MEDIASUBTYPE\_Y41P **Y41P**格式，以4:1:1方式打包

MEDIASUBTYPE\_Y411 **Y411**格式（实际格式与Y41P相同）

MEDIASUBTYPE\_Y211 **Y211**格式

MEDIASUBTYPE\_IF09 IF09格式

MEDIASUBTYPE\_IYUV IYUV格式

MEDIASUBTYPE\_YV12 YV12格式

MEDIASUBTYPE\_YVU9 YVU9格式

               表2.3

**YUV 采样**

YUV 的优点之一是，色度频道的采样率可比 Y 频道低，同时不会明显降低视觉质量。有一种表示法可用来描述 U 和 V 与 Y 的采样频率比例，这个表示法称为 A:B:C 表示法：

|  |  |
| --- | --- |
| • | 4:4:4 表示色度频道没有下采样。 |
| • | 4:2:2 表示 2:1 的水平下采样，没有垂直下采样。对于每两个 U 样例或 V 样例，每个扫描行都包含四个 Y 样例。 |
| • | 4:2:0 表示 2:1 的水平下采样，2:1 的垂直下采样。 |
| • | 4:1:1 表示 4:1 的水平下采样，没有垂直下采样。对于每个 U 样例或 V 样例，每个扫描行都包含四个 Y 样例。与其他格式相比，4:1:1 采样不太常用，本文不对其进行详细讨论。 |

图 1 显示了 4:4:4 图片中使用的采样网格。灯光样例用叉来表示，色度样例则用圈表示。

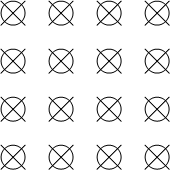


图 1. YUV 4:4:4 样例位置

4:2:2 采样的这种主要形式在 ITU-R Recommendation BT.601 中进行了定义。图 2 显示了此标准定义的采样网格。

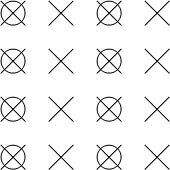


图 2. YUV 4:2:2 样例位置

4:2:0 采样有两种常见的变化形式。其中一种形式用于 MPEG-2 视频，另一种形式用于 MPEG-1 以及 ITU-T recommendations H.261 和 H.263。图 3 显示了 MPEG-1 方案中使用的采样网格，图 4 显示了 MPEG-2 方案中使用的采样网格。

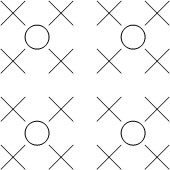


图 3. YUV 4:2:0 样例位置（MPEG-1 方案）

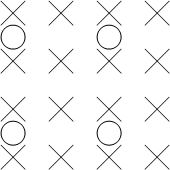


图 4. YUV 4:2:0 样例位置（MPEG-2 方案）

与 MPEG-1 方案相比，在 MPEG-2 方案与为 4:2:2 和 4:4:4 格式定义的采样网格之间进行转换更简单一些。因此，在 Windows 中首选 MPEG-2 方案，应该考虑将其作为 4:2:0 格式的默认转换方案。

**表面定义**

本节讲述推荐用于视频呈现的 8 位 YUV 格式。这些格式可以分为几个类别：

|  |  |
| --- | --- |
| • | 4:4:4 格式，每像素 32 位 |
| • | 4:2:2 格式，每像素 16 位 |
| • | 4:2:0 格式，每像素 16 位 |
| • | 4:2:0 格式，每像素 12 位 |

首先，您应该理解下列概念，这样才能理解接下来的内容：

|  |  |
| --- | --- |
| • | **表面原点**。对于本文讲述的 YUV 格式，原点 (0,0) 总是位于表面的左上角。 |
| • | **跨距**。表面的跨距，有时也称为间距，指的是表面的宽度，以字节数表示。对于一个表面原点位于左上角的表面来说，跨距总是正数。 |
| • | **对齐**。表面的对齐是根据图形显示驱动程序的不同而定的。表面始终应该 DWORD 对齐，就是说，表面中的各个行肯定都是从 32 位 (DWORD) 边界开始的。对齐可以大于 32 位，但具体取决于硬件的需求。 |
| • | **打包格式与平面格式**。YUV 格式可以分为打包 格式和平面 格式。在打包格式中，Y、U 和 V 组件存储在一个数组中。像素被组织到了一些巨像素组中，巨像素组的布局取决于格式。在平面格式中，Y、U 和 V 组件作为三个单独的平面进行存储。 |

**4:4:4 格式，每像素 32 位**

推荐一个 4:4:4 格式，FOURCC 码为 AYUV。这是一个打包格式，其中每个像素都被编码为四个连续字节，其组织顺序如下所示。

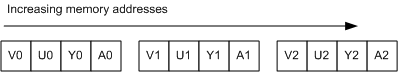


图 5. AYUV 内存布局

标记了 A 的字节包含 alpha 的值。

**4:2:2 格式，每像素 16 位**

支持两个 4:2:2 格式，FOURCC 码如下：

|  |  |
| --- | --- |
| • | YUY2 |
| • | UYVY |

两个都是打包格式，其中每个巨像素都是编码为四个连续字节的两个像素。这样会使得色度水平下采样乘以系数 2。

**YUY2**

在 YUY2 格式中，数据可被视为一个不带正负号的 **char** 值组成的数组，其中第一个字节包含第一个 Y 样例，第二个字节包含第一个 U (Cb) 样例，第三个字节包含第二个 Y 样例，第四个字节包含第一个 V (Cr) 样例，如图 6 所示。

http://www.microsoft.com/china/MSDN/library/enterprisedevelopment/softwaredev/art/yuvformats02.gif

图 6. YUY2 内存布局

如果该图像被看作由两个 little-endian **WORD** 值组成的数组，则第一个 **WORD** 在最低有效位 (LSB) 中包含 Y0，在最高有效位 (MSB) 中包含 U。第二个 **WORD** 在 LSB 中包含 Y1，在 MSB 中包含 V。

YUY2 是用于 Microsoft DirectX® Video Acceleration (DirectX VA) 的首选 4:2:2 像素格式。预期它会成为支持 4:2:2 视频的 DirectX VA 加速器的中期要求。

**UYVY**

此格式与 YUY2 相同，只是字节顺序是与之相反的 — 就是说，色度字节和灯光字节是翻转的（图 7）。如果该图像被看作由两个 little-endian **WORD** 值组成的数组，则第一个 **WORD** 在 LSB 中包含 U，在 MSB 中包含 Y0，第二个 **WORD** 在 LSB 中包含 V，在 MSB 中包含 Y1。

http://www.microsoft.com/china/MSDN/library/enterprisedevelopment/softwaredev/art/yuvformats03.gif

图 7. UYVY 内存布局

**4:2:0 格式，每像素 16 位**

推荐两个 4:2:0 每像素 16 位格式，FOURCC 码如下：

|  |  |
| --- | --- |
| • | IMC1 |
| • | IMC3 |

两个 FOURCC 码都是平面格式。色度频道在水平方向和垂直方向上都要以系数 2 来进行再次采样。

**IMC1**

所有 Y 样例都会作为不带正负号的 **char** 值组成的数组首先显示在内存中。后面跟着所有 V (Cr) 样例，然后是所有 U (Cb) 样例。V 和 U 平面与 Y 平面具有相同的跨距，从而生成如图 8 所示的内存的未使用区域。

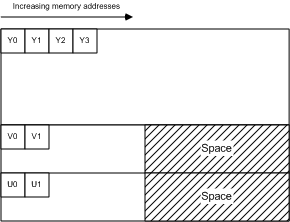


图 8. IMC1 内存布局

**IMC3**

此格式与 IMC1 相同，只是 U 和 V 平面进行了交换：

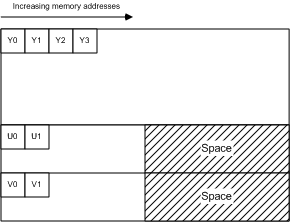


图 9. IMC3 内存布局

**4:2:0 格式，每像素 12 位**

推荐四个 4:2:0 每像素 12 位格式，FOURCC 码如下：

|  |  |
| --- | --- |
| • | IMC2 |
| • | IMC4 |
| • | YV12 |
| • | NV12 |

在所有这些格式中，色度频道在水平方向和垂直方向上都要以系数 2 来进行再次采样。

**IMC2**

此格式与 IMC1 相同，只是 V (Cr) 和 U (Cb) 行在半跨距边界处进行了交错。换句话说，就是色度区域中的每个完整跨距行都以一行 V 样例开始，然后是一行在下一个半跨距边界处开始的 U 样例（图 10）。此布局与 IMC1 相比，能够更加高效地利用地址空间。它的色度地址空间缩小了一半，因此整体地址空间缩小了 25%。在各个 4:2:0 格式中，IMC2 是第二首选格式，排在 NV12 之后。

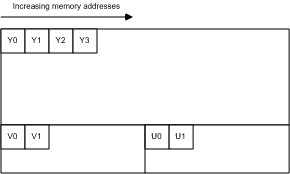


图 10. IMC2 内存布局

**IMC4**

此格式与 IMC2 相同，只是 U (Cb) 和 V (Cr) 行进行了交换：

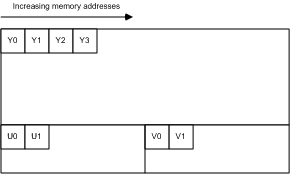


图 11. IMC4 内存布局

**YV12**

所有 Y 样例都会作为不带正负号的 **char** 值组成的数组首先显示在内存中。此数组后面紧接着所有 V (Cr) 样例。V 平面的跨距为 Y 平面跨距的一半，V 平面包含的行为 Y 平面包含行的一半。V 平面后面紧接着所有 U (Cb) 样例，它的跨距和行数与 V 平面相同（图 12）。

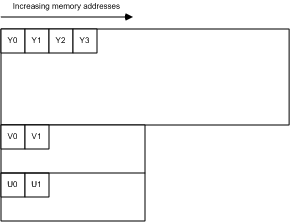
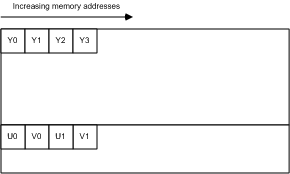


图 12. YV12 内存布局

**NV12**

所有 Y 样例都会作为由不带正负号的 **char** 值组成的数组首先显示在内存中，并且行数为偶数。Y 平面后面紧接着一个由不带正负号的 **char** 值组成的数组，其中包含了打包的 U (Cb) 和 V (Cr) 样例，如图 13 所示。当组合的 U-V 数组被视为一个由 little-endian **WORD** 值组成的数组时，LSB 包含 U 值，MSB 包含 V 值。NV12 是用于 DirectX VA 的首选 4:2:0 像素格式。预期它会成为支持 4:2:0 视频的 DirectX VA 加速器的中期要求。



**YUV格式解析2**

又确认了一下H264的视频格式——H264支持4：2：0的连续或隔行视频的编码和解码

YUV（亦称YCrCb）是被欧洲电视系统所采用的一种颜色编码方法（属于PAL）。YUV主要用于优化彩色视频信号的传输，使其向后兼容老式黑白电视。与RGB视频信号传输相比，它最大的优点在于只需占用极少的带宽（RGB要求三个独立的视频信号同时传输）。其中“Y”表示明亮度（Luminance或Luma），也就是灰阶值；而“U”和“V”表示的则是色度（Chrominance或Chroma），作用是描述影像色彩及饱和度，用于指定像素的颜色。“亮度”是通过RGB输入信号来创建的，方法是将RGB信号的特定部分叠加到一起。“色度”则定义了颜色的两个方面—色调与饱和度，分别用Cr和CB来表示。其中，Cr反映了GB输入信号红色部分与RGB信号亮度值之间的差异。而CB反映的是RGB输入信号蓝色部分与RGB信号亮度值之同的差异。

补充一下场的概念——

场的概念不是从DV才开始有的，电视系统已经有了（当然，DV和电视的关系大家都知道）归根结底还是扫描的问题，具体到PAL制式是：   
每秒25帧，每帧两场，扫描线（包括电视机的电子束）自上而下先扫描一场，然后再自上而下扫描第二场   
之所以引入场的概念，我的理解是主要为了在有限的带宽和成本内使画面运动更加平滑和消除闪烁感。   
这两个场的扫描线是一条一条互相间隔开的，比如说对于一个帧来讲，最上面一条线编号为0，紧挨着的是1，再下来是2，3，4，5，6。。。。那么第一场也许是0，2，4，6；也许是1，3，5，7——这就是隔行扫描   
在逐行扫描模式下，就是扫描线按照0，1，2，3，4，5的顺序依次扫描，很明显，这时候就不存在场的概念了。

下面区分一下YUV和YCbCr

YUV色彩模型来源于RGB模型，

该模型的特点是将亮度和色度分离开，从而适合于图像处理领域。

应用：模拟领域

Y'= 0.299\*R' + 0.587\*G' + 0.114\*B'

U'= -0.147\*R' - 0.289\*G' + 0.436\*B' = 0.492\*(B'- Y')

V'= 0.615\*R' - 0.515\*G' - 0.100\*B' = 0.877\*(R'- Y')

R' = Y' + 1.140\*V'

G' = Y' - 0.394\*U' - 0.581\*V'

B' = Y' + 2.032\*U'

YCbCr模型来源于YUV模型。YCbCr是 YUV 颜色空间的偏移版本.

应用：数字视频，ITU-R BT.601建议

Y’ = 0.257\*R' + 0.504\*G' + 0.098\*B' + 16

Cb' = -0.148\*R' - 0.291\*G' + 0.439\*B' + 128

Cr' = 0.439\*R' - 0.368\*G' - 0.071\*B' + 128

R' = 1.164\*(Y’-16) + 1.596\*(Cr'-128)

G' = 1.164\*(Y’-16) - 0.813\*(Cr'-128) - 0.392\*(Cb'-128)

B' = 1.164\*(Y’-16) + 2.017\*(Cb'-128)

PS: 上面各个符号都带了一撇，表示该符号在原值基础上进行了伽马校正,伽马校正有助于弥补在抗锯齿的过程中，线性分配伽马值所带来的细节损失，使图像细节更加丰富。在没有采用伽马校正的情况下，暗部细节不容易显现出来，而采用了这一图像增强技术以后，图像的层次更加明晰了。

所以说H264里面的YUV应属于YCbCr.

下面再仔细谈谈YUV格式, YUV格式通常有两大类：打包（packed）格式和平面（planar）格式。前者将YUV分量存放在同一个数组中，通常是几个相邻的像素组成一个宏像素（macro-pixel）；而后者使用三个数组分开存放YUV三个分量，就像是一个三维平面一样。

我们常说得YUV420属于planar格式的YUV, 颜色比例如下:

Y0U0V0             Y1                 Y2U2V2                      Y3

Y4                 Y5                 Y6                          Y7

Y8U8V8             Y9                 Y10U10V10                   Y11

Y12                Y13                Y14                         Y15

其他格式YUV可以点这里查看详细内容, 而在YUV文件中YUV420又是怎么存储的呢? 在常见H264测试的YUV序列中,例如CIF图像大小的YUV序列(352\*288),在文件开始并没有文件头,直接就是YUV数据,先存第一帧的Y信息,长度为352\*288个byte, 然后是第一帧U信息长度是352\*288/4个byte, 最后是第一帧的V信息,长度是352\*288/4个byte, 因此可以算出第一帧数据总长度是352\*288\*1.5,即152064个byte, 如果这个序列是300帧的话, 那么序列总长度即为152064\*300=44550KB,这也就是为什么常见的300帧CIF序列总是44M的原因.

4:4:4采样就是说三种元素Y,Cb,Cr有同样的分辨率,这样的话,在每一个像素点上都对这三种元素进行采样.数字4是指在水平方向上对于各种元素的采样率,比如说,每四个亮度采样点就有四个Cb的Cr采样值.4:4:4采样完整地保留了所有的信息值.4:2:2采样中(有时记为YUY2),色度元素在纵向与亮度值有同样的分辨率,而在横向则是亮度分辨率的一半(4:2:2表示每四个亮度值就有两个Cb和Cr采样.)4:2:2视频用来构造高品质的视频彩色信号.

在流行的4:2:0采样格式中(常记为**YV12**)Cb和Cr在水平和垂直方向上有Y分辨率的一半.4:2:0有些不同，因为它并不是指在实际采样中使用4:2:0，而是在编码史中定义这种编码方法是用来区别于4:4:4和4:2:2方法的).4:2:0采样被广泛地应用于消费应用中，比如视频会议，数字电视和DVD存储中。因为每个颜色差别元素中包含了四分之一的Y采样元素量，那么4:2:0YCbCr视频需要刚好4: 4:4或RGB视频中采样量的一半。

4:2:0采样有时被描述是一个"每像素12位"的方法。这么说的原因可以从对四个像素的采样中看出. 使用4:4:4采样，一共要进行12次采样，对每一个Y,Cb和Cr，就需要12\*8=96位，平均下来要96/4=24位。使用4:2:0就需要6\*8 =48位，平均每个像素48/4=12位。

在一个4:2:0隔行扫描的视频序列中，对应于一个完整的视频帧的Y,Cb,Cr采样分配到两个场中。可以得到，隔行扫描的总采样数跟渐进式扫描中使用的采样数目是相同的。

对比一下：

Y41P（和Y411）（packed格式）格式为每个像素保留Y分量，而UV分量在水平方向上每4个像素采样一次。一个宏像素为12个字节，实际表示8个像素。图像数据中YUV分量排列顺序如下： U0 Y0 V0 Y1 U4 Y2 V4 Y3 Y4 Y5 Y6 Y8 …

IYUV格式（planar）为每个像素都提取Y分量，而在UV分量的提取时，首先将图像分成若干个2 x 2的宏块，然后每个宏块提取一个U分量和一个V分量。YV12格式与IYUV类似，但仍然是平面模式。

YUV411、YUV420格式多见于DV数据中，前者用于NTSC制，后者用于PAL制。YUV411为每个像素都提取Y分量，而UV分量在水平方向上每4个像素采样一次。YUV420并非V分量采样为0，而是跟YUV411相比，在水平方向上提高一倍色差采样频率，在垂直方向上以U/V间隔的方式减小一半色差采样，如下图所示。

（好像显示不出来突下图像）

各种格式的具体使用位数的需求（使用4:2:0采样，对于每个元素用8个位大小表示)：

格式： Sub-QCIF 亮度分辨率： 128\*96   每帧使用的位: 147456  
格式： QCIF   亮度分辨率： 176\*144   每帧使用的位: 304128  
格式： CIF   亮度分辨率： 352\*288   每帧使用的位: 1216512  
格式： 4CIF   亮度分辨率： 704\*576   每帧使用的位: 4866048