

# Single-planar format structure

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master] [Documentation] [userspace-api] [media] [v4l]pixfmt-v4l2.rst, line 7)**

Unknown directive type "tabularcolumns".

```
.. tabularcolumns:: |p{4.0cm}|p{2.6cm}|p{10.7cm}|
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master] [Documentation] [userspace-api] [media] [v4l]pixfmt-v4l2.rst, line 9)**

Unknown directive type "c.type".

```
.. c:type:: v4l2_pix_format
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master] [Documentation] [userspace-api] [media] [v4l]pixfmt-v4l2.rst, line 11)**

Unknown directive type "cssclass".

```
.. cssclass:: longtable
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master] [Documentation] [userspace-api] [media] [v4l]pixfmt-v4l2.rst, line 13)**

Unknown directive type "flat-table".

```
.. flat-table:: struct v4l2_pix_format
:header-rows: 0
:stub-columns: 0
:widths:      1 1 2
```

- \* - `__u32`  
- `__width``  
- Image width in pixels.
- \* - `__u32`  
- `__height``  
- Image height in pixels. If `__field`` is one of `__V4L2_FIELD_TOP``, `__V4L2_FIELD_BOTTOM`` or `__V4L2_FIELD_ALTERNATE`` then height refers to the number of lines in the field, otherwise it refers to the number of lines in the frame (which is twice the field height for interlaced formats).
- \* - `:cspan:2`` Applications set these fields to request an image size, drivers return the closest possible values. In case of planar formats the `__width`` and `__height`` applies to the largest plane. To avoid ambiguities drivers must return values rounded up to a multiple of the scale factor of any smaller planes. For example when the image format is YUV 4:2:0, `__width`` and `__height`` must be multiples of two.

For compressed formats that contain the resolution information encoded inside the stream, when fed to a stateful mem2mem decoder, the fields may be zero to rely on the decoder to detect the right values. For more details see :ref:`decoder` and format descriptions.

For compressed formats on the CAPTURE side of a stateful mem2mem encoder, the fields must be zero, since the coded size is expected to be calculated internally by the encoder itself, based on the OUTPUT side. For more details see :ref:`encoder` and format descriptions.

- \* - `__u32`  
- `__pixelformat``  
- The pixel format or type of compression, set by the application. This is a little endian  
:ref:`four character code <v4l2-fourcc>`. V4L2 defines standard RGB formats in :ref:`pixfmt-rgb`, YUV formats in :ref:`yuv-formats`, and reserved codes in

- `:ref:`reserved-formats``
- \* - `u32`
- ```field```
- Field order, from enum `:c:type:`v4l2_field``.  
Video images are typically interlaced. Applications can request to capture or output only the top or bottom field, or both fields interlaced or sequentially stored in one buffer or alternating in separate buffers. Drivers return the actual field order selected. For more details on fields see `:ref:`field-order``.
- \* - `u32`
- ```bytesperline```
- Distance in bytes between the leftmost pixels in two adjacent lines.
- \* - `:cspan:`2``

Both applications and drivers can set this field to request padding bytes at the end of each line. Drivers however may ignore the value requested by the application, returning ```width``` times bytes per pixel or a larger value required by the hardware. That implies applications can just set this field to zero to get a reasonable default.

Video hardware may access padding bytes, therefore they must reside in accessible memory. Consider cases where padding bytes after the last line of an image cross a system page boundary. Input devices may write padding bytes, the value is undefined. Output devices ignore the contents of padding bytes.

When the image format is planar the ```bytesperline``` value applies to the first plane and is divided by the same factor as the ```width``` field for the other planes. For example the Cb and Cr planes of a YUV 4:2:0 image have half as many padding bytes following each line as the Y plane. To avoid ambiguities drivers must return a ```bytesperline``` value rounded up to a multiple of the scale factor.

For compressed formats the ```bytesperline``` value makes no sense. Applications and drivers must set this to 0 in that case.

- \* - `u32`
- ```sizeimage```
- Size in bytes of the buffer to hold a complete image, set by the driver. Usually this is ```bytesperline``` times ```height```. When the image consists of variable length compressed data this is the number of bytes required by the codec to support the worst-case compression scenario.

The driver will set the value for uncompressed images.

Clients are allowed to set the `sizeimage` field for variable length compressed data flagged with ```V4L2_FMT_FLAG_COMPRESSED``` at `:ref:`VIDIOC_ENUM_FMT``, but the driver may ignore it and set the value itself, or it may modify the provided value based on alignment requirements or minimum/maximum size requirements. If the client wants to leave this to the driver, then it should set `sizeimage` to 0.

- \* - `u32`
- ```colorspace```
- Image colorspace, from enum `:c:type:`v4l2_colorspace``.  
This information supplements the ```pixelformat``` and must be set by the driver for capture streams and by the application for output streams, see `:ref:`colorspaces``. If the application sets the flag ```V4L2_PIX_FMT_FLAG_SET_CSC``` then the application can set this field for a capture stream to request a specific colorspace for the captured image data. If the driver cannot handle requested conversion, it will return another supported colorspace. The driver indicates that colorspace conversion is supported by setting the flag `V4L2_FMT_FLAG_CSC_COLORSPACE` in the corresponding struct `:c:type:`v4l2_fmtdesc`` during enumeration. See `:ref:`fmtdesc-flags``.
- \* - `u32`
- ```priv```
- This field indicates whether the remaining fields of the struct `:c:type:`v4l2_pix_format``, also called the extended fields, are valid. When set to ```V4L2_PIX_FMT_PRIV_MAGIC```, it indicates that the extended fields have been correctly initialized. When set to any other value it indicates that the extended fields contain undefined values.

Applications that wish to use the pixel format extended fields must first ensure that the feature is supported by querying the device for the `:ref:`V4L2_CAP_EXT_PIX_FORMAT`` <querycap> capability. If the capability isn't set the pixel format extended

fields are not supported and using the extended fields will lead to undefined results.

To use the extended fields, applications must set the ``priv`` field to ``V4L2\_PIX\_FMT\_PRIV\_MAGIC``, initialize all the extended fields and zero the unused bytes of the struct :c:type:`v4l2\_format` ``raw\_data`` field.

When the ``priv`` field isn't set to ``V4L2\_PIX\_FMT\_PRIV\_MAGIC`` drivers must act as if all the extended fields were set to zero. On return drivers must set the ``priv`` field to ``V4L2\_PIX\_FMT\_PRIV\_MAGIC`` and all the extended fields to applicable values.

- \* - `__u32`
- ``flags``
- Flags set by the application or driver, see :ref:`format-flags`.
- \* - union {
- (anonymous)
- \* - `__u32`
- ``ycbcr\_enc``
- Y'CbCr encoding, from enum :c:type:`v4l2\_ycbcr\_encoding`. This information supplements the ``colorspace`` and must be set by the driver for capture streams and by the application for output streams, see :ref:`colorspaces`. If the application sets the flag ``V4L2\_PIX\_FMT\_FLAG\_SET\_CSC`` then the application can set this field for a capture stream to request a specific Y'CbCr encoding for the captured image data. If the driver cannot handle requested conversion, it will return another supported encoding. This field is ignored for HSV pixelformats. The driver indicates that ycbcr\_enc conversion is supported by setting the flag V4L2\_FMT\_FLAG\_CSC\_YCBCR\_ENC in the corresponding struct :c:type:`v4l2\_fmtdesc` during enumeration. See :ref:`fmtdesc-flags`.
- \* - `__u32`
- ``hsv\_enc``
- HSV encoding, from enum :c:type:`v4l2\_hsv\_encoding`. This information supplements the ``colorspace`` and must be set by the driver for capture streams and by the application for output streams, see :ref:`colorspaces`. If the application sets the flag ``V4L2\_PIX\_FMT\_FLAG\_SET\_CSC`` then the application can set this field for a capture stream to request a specific HSV encoding for the captured image data. If the driver cannot handle requested conversion, it will return another supported encoding. This field is ignored for non-HSV pixelformats. The driver indicates that hsv\_enc conversion is supported by setting the flag V4L2\_FMT\_FLAG\_CSC\_HSV\_ENC in the corresponding struct :c:type:`v4l2\_fmtdesc` during enumeration. See :ref:`fmtdesc-flags`.
- \* - }
- 
- \* - `__u32`
- ``quantization``
- Quantization range, from enum :c:type:`v4l2\_quantization`. This information supplements the ``colorspace`` and must be set by the driver for capture streams and by the application for output streams, see :ref:`colorspaces`. If the application sets the flag ``V4L2\_PIX\_FMT\_FLAG\_SET\_CSC`` then the application can set this field for a capture stream to request a specific quantization range for the captured image data. If the driver cannot handle requested conversion, it will return another supported quantization. The driver indicates that quantization conversion is supported by setting the flag V4L2\_FMT\_FLAG\_CSC\_QUANTIZATION in the corresponding struct :c:type:`v4l2\_fmtdesc` during enumeration. See :ref:`fmtdesc-flags`.
- \* - `__u32`
- ``xfer\_func``
- Transfer function, from enum :c:type:`v4l2\_xfer\_func`. This information supplements the ``colorspace`` and must be set by the driver for capture streams and by the application for output streams, see :ref:`colorspaces`. If the application sets the flag ``V4L2\_PIX\_FMT\_FLAG\_SET\_CSC`` then the application can set this field for a capture stream to request a specific transfer function for the captured image data. If the driver cannot handle requested conversion, it will return another supported transfer function. The driver indicates that xfer\_func conversion is supported by setting the flag V4L2\_FMT\_FLAG\_CSC\_XFER\_FUNC in the corresponding struct :c:type:`v4l2\_fmtdesc` during enumeration. See :ref:`fmtdesc-flags`.

Unknown directive type "tabularcolumns".

```
.. tabularcolumns:: |p{6.8cm}|p{2.3cm}|p{8.2cm}|
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\linux-master [Documentation] [userspace-api] [media] [v4l]pixfmt-v4l2.rst, line 212)**

Unknown directive type "flat-table".

```
.. flat-table:: Format Flags
   :header-rows: 0
   :stub-columns: 0
   :widths:      3 1 4

* - ``V4L2_PIX_FMT_FLAG_PREMUL_ALPHA``
  - 0x00000001
  - The color values are premultiplied by the alpha channel value. For
    example, if a light blue pixel with 50% transparency was described
    by RGBA values (128, 192, 255, 128), the same pixel described with
    premultiplied colors would be described by RGBA values (64, 96,
    128, 128)
* .. _v4l2-pix-fmt-flag-set-csc:

  - ``V4L2_PIX_FMT_FLAG_SET_CSC``
  - 0x00000002
  - Set by the application. It is only used for capture and is
    ignored for output streams. If set, then request the device to do
    colorspace conversion from the received colorspace to the requested
    colorspace values. If the colorimetry field (``colorspace``, ``xfer_func``,
    ``ycbcr_enc``, ``hsv_enc`` or ``quantization``) is set to ``*_DEFAULT``,
    then that colorimetry setting will remain unchanged from what was received.
    So in order to change the quantization, only the ``quantization`` field shall
    be set to non default value (``V4L2_QUANTIZATION_FULL_RANGE`` or
    ``V4L2_QUANTIZATION_LIM_RANGE``) and all other colorimetry fields shall
    be set to ``*_DEFAULT``.

    To check which conversions are supported by the hardware for the current
    pixel format, see :ref:`fmtdesc-flags`.
```