**DPX Application Note for DSC Software Model**

This document describes issues related to DPX files and the DSC software model. Many common implementations of the DPX file format do not follow the document SMPTE 268M.

The DPX file format is a general way of storing picture data for images and video. A header defines the characteristics of each picture, and the pixel data is represented after the header. A magic number in the header is intended to be used to determine the endianness in which the file has been written. There are several things that are ambiguous or frequently misinterpreted in the DPX spec:

1. In some implementations of DPX, the endianness of the DPX header applies to the image data; in other implementations, the image data is always assumed to be big-endian. There is no indication in section 3.1 of SMPTE 268M which interpretation is correct. Annex B (informative) of the specification seems to imply that the detected endianness should apply to all the component data, and Annex C (informative) seems to imply the opposite is true.

**To configure the reader to follow the endianness of the magic number, set DPX\_READ\_FORCE\_BE to 0. To configure the reader to always use big-endian order for the image data (default), set DPXR\_FORCE\_BE to 1. For writing files, the model auto-selects the endianness of the magic number and the header so that the data is written big-endian.**

1. In some implementations of DPX, the component ordering of RGB data is red followed by green followed by blue. Note 2 in Table 1 of SMPTE 268M specifies that the ordering of RGB components is blue followed by green followed by red.

**To configure the reader to use the SMPTE 268M order (BGR, default), set SWAP\_R\_AND\_B to 0. To configure the reader to use the RGB order, set SWAP\_R\_AND\_B to 1. For writing DPX files, configure SWAP\_R\_AND\_B\_OUT to either 0 (BGR) or to 1 (RGB).**

1. In some implementations of DPX, the datum ordering is sequential (following the file order). Annex C (informative) of SMPTE 268M indicates that the first datum is located the least significant *n*bits of the 32-bit words.

**To configure the reader to use the SMPTE 268M order (default), set DPXR\_DATUM\_ORDER to 0. To configure the reader to use the sequential order, set DPXR\_DATUM\_ORDER to 1. For writing files, configure DPXW\_DATUM\_ORDER as appropriate**

1. In some implementations of DPX, “filling” or padding using zeroes is done to 32-bit boundaries at the end of each scan line in 8-bit modes. Annex C (informative) of SMPTE 268M seems to indicate that filling is only necessary if the non-filled, non-aligned 10- or 12-bit formats are used.

**To configure the reader to use the SMPTE 268M filling rules (default), set DPXR\_PAD\_ENDS to 0. To configure the reader to fill scan lines to 32-bit words regardless of mode, set DPXR\_PAD\_8B\_ENDS to 1. For writing files, configure DPXW\_PAD\_ENDS as appropriate.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tool & format** | **(1)** | **(2)** | **(3)** | **(4)** |
| **XnView default** | 1  0 | 0  0 | 1  0 | 1  1 |
| **XnView MSBF option** | 1 | 0 | 1 | 1 |
| **XnView 10 bit/cpnt option** | 0 (1\*)  0 (1\*) | 0  1 | 0  1 | X |
| **XnView 10 bit/cpnt & MSBF options** | 1  1  0  0 | 0  1  0  1 | 0  1  0  1 | X |
| **GraphicsMagick 8 bit/cpnt** | X  X | 1  1 | 1  0 | 1  1 |
| **GraphicsMagick 10 bit/cpnt** | X  X | 0  1 | 0  1 | X |
| **GraphicsMagick 12 bit/cpnt** | X | 1 | 1 | X |
| **GraphicsMagick 16 bit/cpnt** | X | 1 | 1 | X |
| **GraphicsMagick 8 bit/cpnt YCbCr** | X | X | 1 | 1 |
| **GraphicsMagick 10 bit/cpnt YCbCr** | X | X | 1 | X |
| **GraphicsMagick 12 bit/cpnt YCbCr** | X | X | 1 | X |
| **GraphicsMagick 16 bit/cpnt YCbCr** | X | X | 1 | X |
| **GraphicsMagick 8 bit/cpnt YCbCr 4:2:2** | X | X | 1 | 1 |
| **GraphicsMagick 10 bit/cpnt YCbCr 4:2:2 [[1]](#footnote-1)** | X | X | 1 | X |
| **GraphicsMagick 12 bit/cpnt YCbCr 4:2:2** | X | X | 1 | X |
| **GraphicsMagick 16 bit/cpnt YCbCr 4:2:2** | X | X | 1 | X |
| **ImageMagick 8 bit/cpnt** | X  X | 1  1 | 1  0 | 1  1 |
| **ImageMagick 10 bit/cpnt** | X  X | 0  1 | 0  1 | X |
| **ImageMagick 12 bit/cpnt** | X | 1 | 1 | X |
| **ImageMagick 16 bit/cpnt** | X | 1 | 1 | X |
| **ImageMagick 8 bit/cpnt YCbCr** | X | X | 1 | 1 |
| **ImageMagick 10 bit/cpnt YCbCr** | X | X | 1 | X |
| **ImageMagick 12 bit/cpnt YCbCr** | X | X | 1 | X |
| **ImageMagick 16 bit/cpnt YCbCr** | X | X | 1 | X |
| **ImageMagick 8 bit/cpnt YCbCr 4:2:2 [[2]](#footnote-2)** | X | X | 1 | 1 |
| **ImageMagick 10 bit/cpnt YCbCr 4:2:2** | X | X | 1 | X |
| **ImageMagick 12 bit/cpnt YCbCr 4:2:2** | X | X | 1 | X |
| **ImageMagick 16 bit/cpnt YCbCr 4:2:2** | X | X | 1 | X |

**Note: The model will try to auto-detect certain types of files for reading (V1.0 files or files generated by previous versions of the DSC C model), override the settings, and process them correctly.**

To verify that the correct read mode is selected, set the PPM\_FILE\_OUTPUT parameter to a “1”, and the C model should output PPM files that can be verified with an image viewer program. PPM format is more interoperable and will render correctly across different viewers and platforms, but can only represent RGB 4:4:4 data.

1. Widths that are not evenly divisible by 4 are not processed correctly [↑](#footnote-ref-1)
2. These files can’t be read because IM declares them as 4:4:4 even though there is 4:2:2-formatted data in the body [↑](#footnote-ref-2)