SMPTE 268M-2003 Specification Inadequacies

Bob Friesenhahn <bfriesen@GraphicsMagick.org>
GraphicsMagick Group
December 3, 2005

Background

The SMPTE 268M specification ("DPX") was first standardized in 1994 and was inspired by a Kodak draft specification for a format ("Cineon") originally (1990) used by the Kodak Cineon System. In spite of its age, DPX has held up well due to using a structured format which supports efficient word-aligned access, supports both video and film oriented data, uses a header with attributes at pre-defined offsets, and allows the file to be updated without being completely re-written (most file formats require the file to be entirely re-written due to almost any change). DPX is the format most often used for 'telecine' applications and for computer-based Digital Intermediate work. There are many thousands of terabytes of DPX files currently in existence around the word.

In spite of the DPX format's apparent success, there have been interoperability problems. Many of these interoperability problems are due to a difference in interpretation of the DPX specification due to an inadequate specification of the pixel storage format. A casual inspection of the specification suggests that the pixel storage format is adequately specified but format implementers soon learn that this is not the case, and in fact some parts of the specification are grossly misleading. Unfortunately, considerable ambiguity was added by the 2003 version of the specification.

The core issues that the DPX specification fails to adequately address are:

- Storage word size (8, 16, 32, or 64 bits)
- Byte order conventions
- Row justification/padding rules
- YCbCr datum ordering of 10-bit components in a 32-bit word (Fig C.3)

Storage Word Size

The DPX specification does not provide a complete specification for storage word sizes. The specification is adequate for the 10-bit packed (Fig C.2), 12-bit packed (Fig C.4), and 10-bit filled (Fig C.3) cases. The specification is clearly inadequate for 8-bit components (Fig C.1). Annex C provides informative data packing diagrams which suggest (but do not specify!) the use of 32-bit words in all cases. Based on the diagrams, it could be assumed that 16-bit words are packed into 32-bit words. It could also be assumed that 8-bit components are packed into 32-bit words using a similar algorithm to that used for the 10-bit and 12-bit packed cases.

The note associated with Table 3B clearly specifies the use of 32 bit words across the board even though it is possible that the 2003 version of the specification intended to handle 16-bit words as an array of 16-bit words.

> Real-world Note: 8 and 16-bit DPX files written by the Kodak Cineon System are indeed written with samples packed into 32-bit words. This differs from files written by the modern Thompson/GrassValley Spirit 4K datacine which write 8-bit files with octets in ascending file offset order, and 16-bit files with 16-bit words in ascending file offset order. The FilmLight NorthLight scanner produces 16-bit files similar to the Spirit 4K. The result is horizontal distortion and wrong colors when a file is read by a system using a different convention.

Byte Order Conventions

Annex B “Byte Order Conventions” provides a justification for why the DPX format supports both big (“Motorola 68K”) and little endian (“Intel x86”) conventions but it fails to define how byte order conventions are applied to the pixel data. The implementer is left to assume that 16 and 32 bit words may need to be “swapped” according to the conventions of the host CPU. Byte order conventions are closely tied to native storage word size. As an example, on a big-endian CPU, the first octet encountered in the file becomes the most significant octet of a 32-bit word. For the 8-bit case, this means that the first octet encountered becomes 'datum 3' in the data packing diagram if the storage size is a 32-bit word, but if the octets are read in ascending file order, then the first octet encountered is 'datum 0'. A similar case occurs for 16-bit components. If 16-bit components are packed into 32-bit words, then 'datum 1' is the first 16-bit quantity read from the file.

The byte order convention figures are misleading in that the figures do not all mean the same thing. The description of the figures says “*These diagrams illustrate the packing of 8-, 10-, 12-, and 16-bit components into 32-bit and 16-bit words, using the most significant- byte first convention*.” Figures C.5 and C.6 are drawn using the same conventions (32-bit diagram) as the other cases even though they could be assumed to use native 16-bit words. The mention of 16-bit words was added in the 2003 version of the specification. Before the 2003 specification there was never mention of anything but a 32-bit word.

In order to solve the byte order convention issue, the standard needs to clearly identify the storage word size for each case so that conventional byte order swapping can occur. If some use 32 bits while others use 16 bits, this needs to be clearly indicated.

> Real world note: The description of DPX in O'Reilly's “Encyclopedia Of Graphics File Formats” claims that “*All components must be stored as words using 32-bit boundaries*”.

Row Justification/Padding Rules

The row justification (padding) rules in the 2003-version of the specification are entirely specified by the sentence at the bottom of Table 3c: “Runs will always break at scan line boundaries. Packing will always break to the next 32-bit word at scan-line boundaries.” The common case of 10-bit components filled into a 32-bit word becomes a problem when components for a row do not fill the last word. In this case there is a choice of inserting empty “pad” components and starting the next row on the next full-word, or inserting components from the next row into the current word. If the later approach is taken, then it is much more difficult to decode the file on a row-by-row basis or update just one row. Based on current text in the specification, one must assume that element data is written without regard for rows except for in the two cases specified by Table 3c. A careful reading of the 1994 version of the specification suggests that an editing change in the 2003 version may have limited the row justification to only apply to packed and RLE compressed data when it was originally intended to apply to all cases.

Field 21.13 “End-of-line padding” presumably allows extra padding to be added to the end of rows, but this option is virtually useless since it has a unit of bytes rather than components or bits. I have yet to see a file where this option has been set.

> Real world note: The Kodak Cineon System pads rows out to a 32-bit boundary. This is clear for 8 and 16-bit files. The description of DPX in O'Reilly's “Encyclopedia Of Graphics File Formats” claims that “*Lines are typically padded with zero bits to end on 4-byte boundaries, although that is not a requirement of the DPX standard. Image data is also typically padded with zero bits to end on an 8K block boundary*.” The Thompson/GrassValley Spirit 4K datacine appears to pad the end of rows for 8-bit data, but not for 16-bit data. FilmLight's Northlight never adds row padding.

Due a lack of clarity on the row padding rules, it is common for equipment handling DPX to insist on image widths which assure that row padding rules would never be applied.

YCbCr datum ordering

There seems to be substantial industry confusion regarding YCbCr datum ordering of 10-bit components in a 32-bit word (Fig C.3). It seems that this is due to note 2 of Table 1 which is supposed to apply to RGB data only. Note 2 (added in the 2003 version of the specification) reverses the datum ordering for RGB but only for the case of 10-bit components in a 32-bit word. Products from a number of significant vendors effectively write YCbCr with the order CrYCb rather than the order CbYCr. If it was intended that both RGB and YCbCr should be reversed, then it would have been much easier to renumber the datums in Annex C.