SMPTE STANDARD

Digital Moving-Picture Exchange (DPX) – Format Extensions for High Dynamic Range and Wide Color Gamut

Table of Contents **Page**

Approved December 3, 2018

Foreword

SMPTE (the Society of Motion Picture and Television Engineers) is an internationally-recognized standards developing organization. Headquartered and incorporated in the United States of America, SMPTE has members in over 80 countries on six continents. SMPTE's Engineering Documents, including Standards, Recommended Practices, and Engineering Guidelines, are prepared by SMPTE's Technology Committees. Participation in these Committees is open to all with a bona fide interest in their work. SMPTE cooperates closely with other standards-developing organizations, including ISO, IEC and ITU.

SMPTE Engineering Documents are drafted in accordance with the rules given in its Standards Operations Manual. This SMPTE Engineering Document was prepared by Technology Committee 31FS.

Intellectual Property

At the time of publication no notice had been received by SMPTE claiming patent rights essential to the implementation of this Engineering Document. However, attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. SMPTE shall not be held responsible for identifying any or all such patent rights.

Introduction

ST 268-1:2014 defines DPX version 2.0. This document defines extensions to DPX version 2.0 that provide support for higher brightness, higher dynamic range, and wider color gamut than was possible at the time DPX was created. Additional transfer functions and colorimetric descriptors are provided to support these usages.

1 Scope

This Standard defines high dynamic range (HDR) and wide color gamut (WCG) extensions for the DPX file format. Files that implement this Standard use version number V2.0HDR in the file information header.

2 Conformance Notation

Normative text is text that describes elements of the design that are indispensable or contains the conformance language keywords: "shall", "should", or "may". Informative text is text that is potentially helpful to the user, but not indispensable, and can be removed, changed, or added editorially without affecting interoperability. Informative text does not contain any conformance keywords.

All text in this document is, by default, normative, except: the Introduction, any section explicitly labeled as "Informative" or individual paragraphs that start with "Note:"

The keywords "shall" and "shall not" indicate requirements strictly to be followed in order to conform to the document and from which no deviation is permitted.

The keywords, "should" and "should not" indicate that, among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited.

The keywords "may" and "need not" indicate courses of action permissible within the limits of the document.

The keyword "reserved" indicates a provision that is not defined at this time, shall not be used, and may be defined in the future. The keyword "forbidden" indicates "reserved" and in addition indicates that the provision will never be defined in the future.

A conformant implementation according to this document is one that includes all mandatory provisions ("shall") and, if implemented, all recommended provisions ("should") as described. A conformant implementation need not implement optional provisions ("may") and need not implement them as described.

Unless otherwise specified, the order of precedence of the types of normative information in this document shall be as follows: Normative prose shall be the authoritative definition; Tables shall be next; then formal languages; then figures; and then any other language forms.

3 Normative References

The following standards contain provisions which, through reference in this text, constitute provisions of this engineering document. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this engineering document are encouraged to investigate the possibility of applying the most recent edition of the standards indicated below.

ISO/IEC 646:1991, Information technology – ISO 70bit coded character set for information interchange

ISO/IEC 10646:2017, Information technology – Universal Coded Character Set (UCS)

IEEE 754-2008, IEEE Standard for Floating-Point Arithmetic

CTA 861-G, CTA Standard, A DTV Profile for Uncompressed High Speed Digital Interfaces (2016)

ISO 11664-3:2012 (CIE S014-3/E:2011), Colorimetry – Part 3: CIE tristimulus values

ISO 16684-1:2012, Graphics technology – Extensible metadata platform (XMP) specification – Part 1: Data model, serialization and core properties

SMPTE ST 12-1:2014, Time and Control Code

SMPTE ST 12-3:2016, Time Code for High Frame Rate Signals and Formatting in the Ancillary Data Space

SMPTE ST 268-1:2014, File Format for Digital Moving-Picture Exchange (DPX)

SMPTE ST 336:2017, Data Encoding Protocol Using Key-Length-Value

SMPTE RP 431-2:2011, D-Cinema Quality – Reference Projector and Environment

SMPTE ST 2001-1:2015, XML Representation of SMPTE Registered Data (Reg-XML) — Mapping Rules

SMPTE ST 2065-1:2012, Academy Color Encoding Specification (ACES)

SMPTE ST 2067-21:2016, Interoperable Master Format – Application #2E

SMPTE ST 2113:2018, Colorimetry of P3 Color Spaces

Recommendation ITU-R BT.2020-2 (10/2015), Parameter values for ultra-high definition television systems for production and international programme exchange

Recommendation ITU-R BT.2100-2 (07/2018), Image parameter values for high dynamic
Note to SMPTE HQ: Please hold publication of 268-2 until ST 2113 is published range television for use in production and international programme exchange

IEC 61966-2-1:1999, Multimedia systems and equipment – Colour measurement and management – Part 2-1: Colour management – Default RGB colour space – sRGB

IEC 61966-2-4:2006, Multimedia systems and equipment – Colour measurement and management – Part 2-4: Colour management – Extended-gamut YCC colour space for video applications – xvYCC

4 Terms and Definitions

The following terms and definitions, in addition to those defined in ST 268-1, apply to this standard.

4.1 extension

field or value that extends the definitions found in ST 268-1

4.2 pixel

smallest addressable constituent of a picture

4.3 sample

value of a single component at a particular location in a picture

4.4 datum

value of a sample within an image element where the bit depth is as indicated in the corresponding image element data structure

4.5 image data word

basic unit for storage of image data (32-bit unsigned integer)

4.6 datum mapping direction

direction that datum values are mapped into image data words

4.7 color-difference subsampling siting

spatial relationship between color difference samples and luminance samples

5 Notation

5.1 Data types

Strings of ASCII characters are notated using a fixed-width font within quotes; for example, "`STRING`". The quote characters do not form part of the value. Strings that do not fill a whole field are terminated with a NUL (zero) character.

Table 1 shall define data types and the corresponding Undefined values.

Table 1 – Data type definitions

Data type	Undefined value	Definition
U8	255 (0xff)	Unsigned 8-bit integer
U16	65535 (0xffff)	Unsigned 16-bit integer
U32	4294967296 (0xffffffff)	Unsigned 32-bit integer
R32	NaN (0xffffffff)	Single-precision binary floating point as defined in IEEE 754
ASCII	"" (empty string)	NUL-terminated ASCII string (as defined in ISO/IEC 646:US)
UTF-8	"" (empty string)	NUL-terminated UTF-8 string (as defined in ISO/IEC 10646)

Note: The definitions for U8, U16, U32, R32, and ASCII are consistent with ST 268-1.

Bits are numbered from least-significant to most-significant, where bit 0 is the least-significant bit of a value. When a direction (left or right) is mentioned in reference to an integer field, the bit(s) to the left are more significant than the bit(s) to the right.

5.2 Core fields and values

The rules defined in section 4.3 of ST 268-1:2014 shall apply. A reader shall interpret all values within fields labeled as "core" with the exception of the image orientation (field 17) and data sign (field 2x.1) fields. For the image orientation and data sign fields, the single value listed as core (i.e., value = 0) shall be interpreted and the other values may be interpreted.

5.3 Field designation tables

A period in the field number in the field designation table indicates that the field is a data structure containing several fields. For example, field 29 is defined as a data structure that comprises fields 29.1 and 29.2.

The offset and length columns are specified in units of bytes. Offsets are specified relative to the beginning of the file.

Core fields (as defined in section 4.3 of ST 268-1:2014) are identified with a ''C'' in the "Core" column of the field designation table.

5.4 Other nomenclature

Fields within the image element data structures (fields 21-28) are sometimes referred to generally with a wildcard “x” that represents that field as it applies to the corresponding image element (for example, “field 2x.9” refers to the bit depth field as it applies to image element x).

The first line in an image element is line 0.

Note: Lines are numbered with respect to the image (pixel) data that is actually represented within the DPX file. Video timing or interface line numbers are not used in this Standard.

Table 2 shall define the operators used in this Standard.

Table 2 – Operator definitions

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
>>	Bitwise right shift
&	Bitwise AND
%	Modulo
$\lfloor a \rfloor$	Floor of a
$\lceil a \rceil$	Ceil of a

6 File

6.1 General structure

A file conforming to this Standard shall contain three or four sections, the first three of which are header information:

- generic file information, image information, data format, and image origination information (fixed length) as defined in section 5 of ST 268-1:2014 along with the extensions defined section 6.2;
- motion-picture and television industry-specific information (fixed length) as defined in section 6 of ST 268-1:2014 along with the extensions defined in section 6.3;
- an optional user-defined data section as defined in section 6.4; and
- image data as defined in section 6.5.

An optional standards-based metadata section may be present at an offset specified in the generic file information header (field 16.1).

The provisions of SMPTE ST 268-1:2014 sections 4.2, 4.3, 4.4, and 4.5 shall apply.

The descriptor (fields 2x.6), datum mapping direction (field 16.2), packing (fields 2x.10), and byte order of the magic number (field 1) shall determine how the datum values are stored in the image data area as defined in section 8 and in Annex A.

The byte order of the magic number shall determine the byte order of the multi-byte fields in header fields as defined in Annex A.

6.2 Generic Header Extensions

6.2.1 File Information Header extensions

Table 3 shall define file information header extensions.

Table 3 – File information header extensions

Field	Offset	Length	Type	Core	Content
1	0	4	U32	C	Magic number. Field 1 shall be equal to 0x53445058 (if read as ASCII, this field reads as "`SDPX`" if the file is in most-significant-byte-first file order or "`XPDS`" in least-significant-byte-first file order).
3	8	8	ASCII	C	Version number. Field 3 shall be equal to "`V2.0HDR`".
6	24	4	U32		Generic section header length in bytes. Field 6 shall be equal to 1664.
7	28	4	U32		Industry-specific header length in bytes. Field 7 shall be equal to 384.
8	32	4	U32		User-defined data length in bytes. Field 8 shall not be Undefined.
16.1	664	4	U32		Standards-based metadata offset; if there is no standards-based metadata section in the file, this field shall have the Undefined value
16.2	668	1	U8	C	Datum mapping direction; as defined in section 7.1
16.3	669	1	U8		Reserved for future use
16.4	670	2	U16		Reserved for future use
16.5	672	96	TBD		Reserved for future use

Note: Previous versions of field 16 were a 104-byte reserved field.

6.2.2 Image Information Header extensions

Table 4 shall define image information header extensions.

The offsets in fields 2x.12 shall be interpreted as byte offsets relative to the beginning of the file.

Table 4 – Image information header extensions

Field	Offset	Length	Type	Core	Content
2x	Data structure for image element x				
2x.2	784	4	U32 or R32	C	Reference low data code value; as defined in SMPTE ST 268-1. For bit depth < 32, the field type shall be U32 and the default value shall be 0; for bit depth >= 32, the field type shall be R32 and the default value shall be 0.0. (Note 1)
2x.4	792	4	U32 or R32	C	Reference high data code value; as defined in SMPTE ST 268-1. For bit depth < 32, the field type shall be U32 and the default value shall be $2^d – 1$, where *d* is equal to the bit depth; for bit depth >= 32, the field type shall be R32 and the default value shall be 1.0. (Note 1)
2x.12	808	4	U32	C	Offset to first image data word. Field 2x.12 shall be a multiple of 4.
2x.13	812	4	U32	C	End-of-line padding in bytes. Field 2x.13 shall be a multiple of 4.
2x.14	816	4	U32		End-of-image padding in bytes. Field 2x.14 shall be a multiple of 4.
29.1	1356	4	U32		Color-difference subsampling siting descriptor; as defined in Table 14
29.2	1360	48	TBD		Reserved for future use

Note 1: Narrow-range and full-range representations, for example, as specified in Recommendation ITU-R BT.2100, can be distinguished by means of the Image Information Header fields 2x.2 (Reference low data code value) and 2x.4 (Reference high data code value). Fields 2x.2, 2x.3 (Reference low quantity represented), 2x.4 and 2x.5 (Reference high quantity represented) can be set in accordance with the values given in the appropriate image standard. For YC_BC_R and IC_TC_P formats where the luminance and color differences use different data code ranges, fields 2x.2 and 2x.4 represent the luminance data code range. These fields are not intended to represent the mastering minimum or peak luminance, or other similar per-image characteristics.

Note 2: Previous versions of field 29 were a 52-byte reserved field.

6.3 Industry-Specific Header Extensions

Table 5 shall define television information header extensions.

Table 5 – Television information header extensions

Field	Offset	Length	Type	Content
58	1920	4	U32	SMPTE time code; as defined in section 7.6
59	1924	4	U32	SMPTE user bits; as defined in section 7.6

74.1	1972	1	U8	Video Identification Code (VIC); as defined in CTA 861-G. If not applicable, the value shall be Undefined.
74.2	1973	1	U8	SMPTE time code type; as defined in Table 15
74.3	1974	1	U8	SMPTE time code DBB2 value; as defined in ST 12-3:2016 section 9.2.2. If not applicable, the value shall be Undefined.
74.4	1975	1	U8	Reserved for future use
74.5	1976	72	TBD	Reserved for future use

Note: Previous versions of field 74 were a 76-byte reserved field.

6.4 User-Defined Data

Table 6 shall define the user-defined data section.

Table 6 – User-defined data

Field	Offset	Length	Type	Content
75	2048	32	ASCII	User identification
76	2080	0 - 1000000	Varies	User defined – Postage stamp, processing logs, etc.

The user-defined data (field 76) provides an extended area for customized information needed by some users. The format and length of this section is not defined by the Standard. If the user-defined data is not present in the file, the value of the user-defined data length (field 8) shall be 0.

6.5 Image Data

Table 7 shall define the image data section.

Table 7 – Image data

Field	Offset	Length	Type	Content
77	As indicated in field 21.12	xx	Sequence of U32	Image data words for image element 1
78	As indicated in field 22.12	xx	Sequence of U32	Image data words for image element 2 (if used)
79	As indicated in field 23.12	xx	Sequence of U32	Image data words for image element 3 (if used)
80	As indicated in field 24.12	xx	Sequence of U32	Image data words for image element 4 (if used)
81	As indicated in field 25.12	xx	Sequence of U32	Image data words for image element 5 (if used)
82	As indicated in field 26.12	xx	Sequence of U32	Image data words for image element 6 (if used)
83	As indicated in field 27.12	xx	Sequence of U32	Image data words for image element 7 (if used)
84	As indicated in field 28.12	xx	Sequence of U32	Image data words for image element 8 (if used)

The offset of the image data shall be as indicated in field 2x.12 and shall be an even multiple of 4.

Image data words stored in the file shall follow the byte order indicated by the magic number (field 1).

The mapping of samples to image data words is a function of the image element descriptor (field 2x.6), the packing method (field 2x.10), and the datum mapping direction (field 16.2), and is described in detail in section 7.2 and section 8.

6.6 Standards-based Metadata

Table 8 shall define the standards-based metadata section. The section (fields 85 through 87) may be located before or after the image data.

Table 8 – Standards-based metadata

Field	Offset	Length	Type	Content
85	S (Note 1)	128	UTF-8	Standards-based metadata format descriptor as defined in section 7.8
86	S+128	4	U32	Standards-based metadata length
87	S+132	Specified by field 86	Array of U8 (Note 2)	Standards-based metadata

Note 1: The value of S is equal to the standards-based metadata offset specified in field 16.1.

Note 2: The data format for the standards-based metadata is specified by the standard referred to by the standards-based metadata format descriptor (field 85). Because field 87 is defined as an Array of U8, the representation of the metadata block in the file is not affected by the magic number (field 1).

7 Field Extensions

7.1 Datum Mapping Direction

Table 9 shall define the datum mapping direction field.

Table 9 (field 16.2) – Datum mapping direction

Value	Description
0	datum values shall be ordered within an image data word starting from the least-significant bit as described in section 8 (right-to-left)
1	datum values shall be ordered within an image data word starting from the most-significant bit as described in section 8 (left-to-right)
2-254	Reserved

The datum mapping direction field affects how datum values are mapped to image data words.

- The field value 0 (right-to-left) places the first datum in the least-significant bits of the first image data word, the second datum in the next-least-significant bits of the first image data word, and so on.
- The field value 1 (left-to-right) places the first datum in the most-significant bits of the first image data word, the second datum in the next-most-significant bits of the first image data word, and so on.

Example illustrations for both directions are provided in Annex B.

Note: The datum mapping direction does not affect the bit order of the datum values. The datum order is specified in section 7.2.2.

7.2 Image Element Descriptors

7.2.1 Image Element Descriptor extensions

Table 10 shall define image element descriptor extensions. Component labels may represent either linear samples (e.g., R, G, B, Y) or nonlinear samples (e.g., R’, G’, B’, Y’) as indicated by the transfer characteristic field.

Table 10 – Image element descriptor extensions

Value	Components (and datum order)
10	Color Difference C_B (4:2:0 color component); as defined in Section 7.2.4
11	Color Difference C_R (4:2:0 color component); as defined in Section 7.2.4
53	B, G, R
54	B, G, R, A
55	A, R, G, B
56	R, G, B
57	R, G, B, A
58	A, B, G, R
104	C (alternating lines of C_B and C_R), Y, Y (4:2:0); as defined in Section 7.2.4
105	C (alternating lines of C_B and C_R), Y, A, Y, A (4:2:0:4); as defined in Section 7.2.4

7.2.2 General provisions related to image element data

Each pixel within an image element shall have 1 to 8 components. All components in an image element shall have the same bit depth, which is specified in field 2x.9. A component value for a pixel is referred to as a datum. The component order specified in Table 10 shall be interpreted as the datum order for each individual pixel. The datum order for the entire image element shall comprise the datum values for each pixel concatenated sequentially in the scan order indicated in field 17 over all of the pixels represented by the current image element data structure. If run-length encoding is used (field 21.11 equal to 1), a run-length count shall appear before each pixel's component data. The run-length count is the same bit depth as a component and specifies the repetition of that pixel. Run-length encoding is described in ST 268-1:2014 Table 3C.

Note: The datum order is not the same as the file byte order, and the mapping of datum order to file byte order is discussed in section 8.

Example: For a value of 53, the first datum (d0) is the Blue component of the first pixel in the scan order indicated in field 17, the second datum (d1) is the Green component of the first pixel, the third datum (d2) is the Red component of the first pixel, the fourth datum (d3) is the Blue component of the second pixel in the scan order indicated in field 17, and so on.

The component values shall not be premultiplied by alpha (A).

An IC_TC_P image shall be packed using any supported format for YC_BC_R component data from Table 10. The I samples shall be represented as the Y component, the C_T samples shall be represented as the C_B component, and the C_P samples shall be represented as the C_R component. All of the rules for YC_BC_R shall apply to IC_TC_P.

7.2.3 Specific provisions related to 4:2:2 image element data

For a 4:2:2 image (i.e., when the image element descriptor is equal to value 7, 100, or 101 as defined in Table 10), the pixels per line value (field 19) shall be an even number and shall represent the number of Y (or I) samples in the image.

Example: In order to represent a 1920x1080 4:2:2 image element, field 19 would be set to 1920.

The two Y components shall represent two horizontally-adjacent pixels from the original image. Therefore, a pixel for the purposes of run-length encoding and component packing shall be two pixels from the same raster line of the original image.

7.2.4 Specific provisions related to 4:2:0 image element data

7.2.4.1 General 4:2:0 provisions

For a 4:2:0 image (i.e., when the image element descriptor is equal to value 10, 11, 104, or 105 as defined in Table 10), the pixels per line value (field 19) and lines per image element value (field 20) shall both be even numbers and shall represent the number of Y (or I) samples in the image.

Example: In order to represent a 1920x1080 4:2:0 image element, field 19 would be set to 1920 and field 20 would be set to 1080.

There are two representations defined for a 4:2:0 image: planar (as described in section 7.2.4.2) and interleaved (as described in section 7.2.4.3).

7.2.4.2 Image element descriptors 10 and 11

For image elements where the image element descriptor is equal to either 10 or 11, the number of samples per line in the image element shall be equal to half of the value in field 19, and the number of lines in the image element shall be equal to half of the value in field 20. A pixel for the purposes of run-length encoding and component packing shall be a single color difference sample.

7.2.4.3 Image element descriptors 104 and 105

For image elements where the image element descriptor is equal to 104 or 105, even-numbered lines contain C_B within the C field and odd-numbered lines contain C_R within the C field. The two Y components shall represent two horizontally-adjacent pixels from the original image. Line 2*m shall contain the C_B samples and line 2*m+1 shall contain the C_R samples, where m is an integer in the range [0, (lines per image element) / 2 – 1]. The C_B and C_R samples are collocated with each other in the original image and the metadata described in Section 7.5 shall indicate the sample siting. An illustration of this datum ordering using a descriptor value of 104 is shown in Figure 1. A pixel for the purposes of run-length encoding and component packing shall be two pixels from the same raster line of the original image.

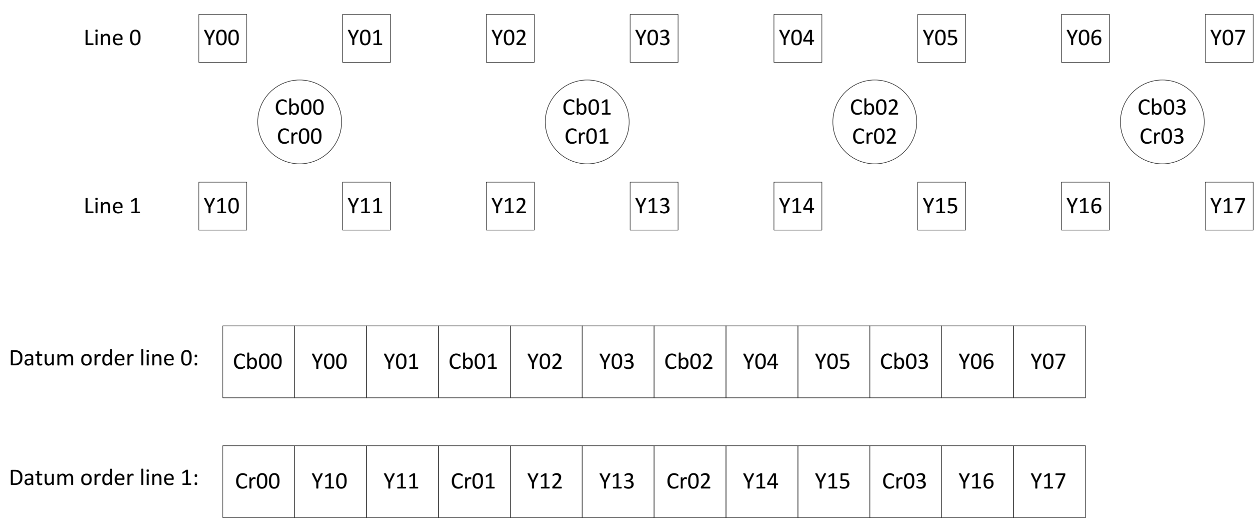

Figure 1 – Example of datum order for a 4:2:0 image element using a descriptor value of 104

7.3 Transfer Characteristic Extensions

Table 11 shall define transfer characteristic extensions.

Table 11 –Transfer characteristic extensions

Value	Transfer Characteristic
14	Recommendation ITU-R BT.2020 non-linear transfer function and non-constant luminance signal format
15	Non-linear transfer function and constant luminance signal format as defined in Recommendation ITU-R BT.2020
16	IEC 61966-2-4 xvYCC
17	Perceptual Quantization (PQ) system reference non-linear transfer functions and non-constant luminance $Y'C'_BC'_R$ signal format as defined in Recommendation ITU-R BT.2100
18	Perceptual Quantization (PQ) system reference non-linear transfer functions and constant intensity IC_TC_P signal format as defined in Recommendation ITU-R BT.2100
19	Hybrid-Log Gamma (HLG) system reference non-linear transfer functions and non-constant luminance $Y'C'_BC'_R$ signal format as defined in Recommendation ITU-R BT.2100
20	Hybrid-Log Gamma (HLG) system reference non-linear transfer functions and constant intensity $\mathrm{IC_TC_P}$ signal format as defined in Recommendation ITU-R BT.2100
21	RP 431-2:2011 Table A1 Gamma 2.6 transfer function
22	IEC 61966-2-1 sRGB

Note: Some of the values in the table indicate signal format as well as transfer characteristic.

7.4 Colorimetric Specification Extensions

Table 12 shall define colorimetric specification extensions.

Table 12 – Colorimetric specification extensions

Value	Colorimetric Specification
14	Recommendation ITU-R BT.2020
15	P3D65; as defined in ST 2113
16	P3DCI; as defined in ST 2113
17	P3D60; as defined in ST 2113
18	ACES color space; as defined in ST 2065-1

Note 1: In ST.268-1:2014 (Table 5A), the numerical value used for Colorimetry is typically the same as the value used for transfer function. For the extensions defined in this Standard (14 and above), this correlation does not apply and transfer function and colorimetric specification can be treated independently.

Note 2: The Recommendation ITU-R BT.709 (value 6) colorimetry can also be used for sRGB and xvYCC colorimetry.

7.5 Color-Difference Subsampling Siting Descriptor

The color-difference subsampling siting for each image element may be indicated using the color-difference subsampling siting descriptor as defined in Table 13. While Table 13 enumerates values for the color-difference subsampling siting for a 4:2:0 image element, values 1 and 2 in Table 13 may also be used to describe the horizontal color-difference subsampling siting for a 4:2:2 image element.

Color differences that are cosited horizontally shall be cosited with even-numbered sample positions. Color differences that are cosited vertically shall be cosited with even-numbered lines. Color differences that are sited interstitially horizontally shall be sited between positions 2*n and 2*n+1 and not between positions 2*n+1 and 2*n+2, where n is a non-negative integer. Color differences that are sited interstitially vertically shall be sited between lines 2*n and 2*n+1 and not between lines 2*n+1 and 2*n+2.

Figure 2 illustrates the relative sample positions of luminance and color-differences for the values in Table 13 for 4:2:0 image elements.

Table 13 – Color-difference subsampling siting bits definition

Value	Color-difference subsampling siting for image element
0	Not applicable (or unknown)
1	Color differences are co-sited with luminance horizontally and vertically
2	Color differences are sited interstitially with luminance horizontally and co-sited vertically
3	Color differences are co-sited with luminance horizontally and sited interstitially vertically
4	Color differences are sited interstitially with luminance horizontally and vertically
5-14	Reserved
15	Undefined

Table 14 (field 29.1) – Color-difference subsampling siting descriptor

Bits	Definition
0 – 3	Color-difference subsampling siting for image element 1 (as defined in Table 13)
4 – 7	Color-difference subsampling siting for image element 2 (as defined in Table 13)
8 - 11	Color-difference subsampling siting for image element 3 (as defined in Table 13)
12 - 15	Color-difference subsampling siting for image element 4 (as defined in Table 13)
16 - 19	Color-difference subsampling siting for image element 5 (as defined in Table 13)
20 – 23	Color-difference subsampling siting for image element 6 (as defined in Table 13)
24 - 27	Color-difference subsampling siting for image element 7 (as defined in Table 13)
28 - 31	Color-difference subsampling siting for image element 8 (as defined in Table 13)

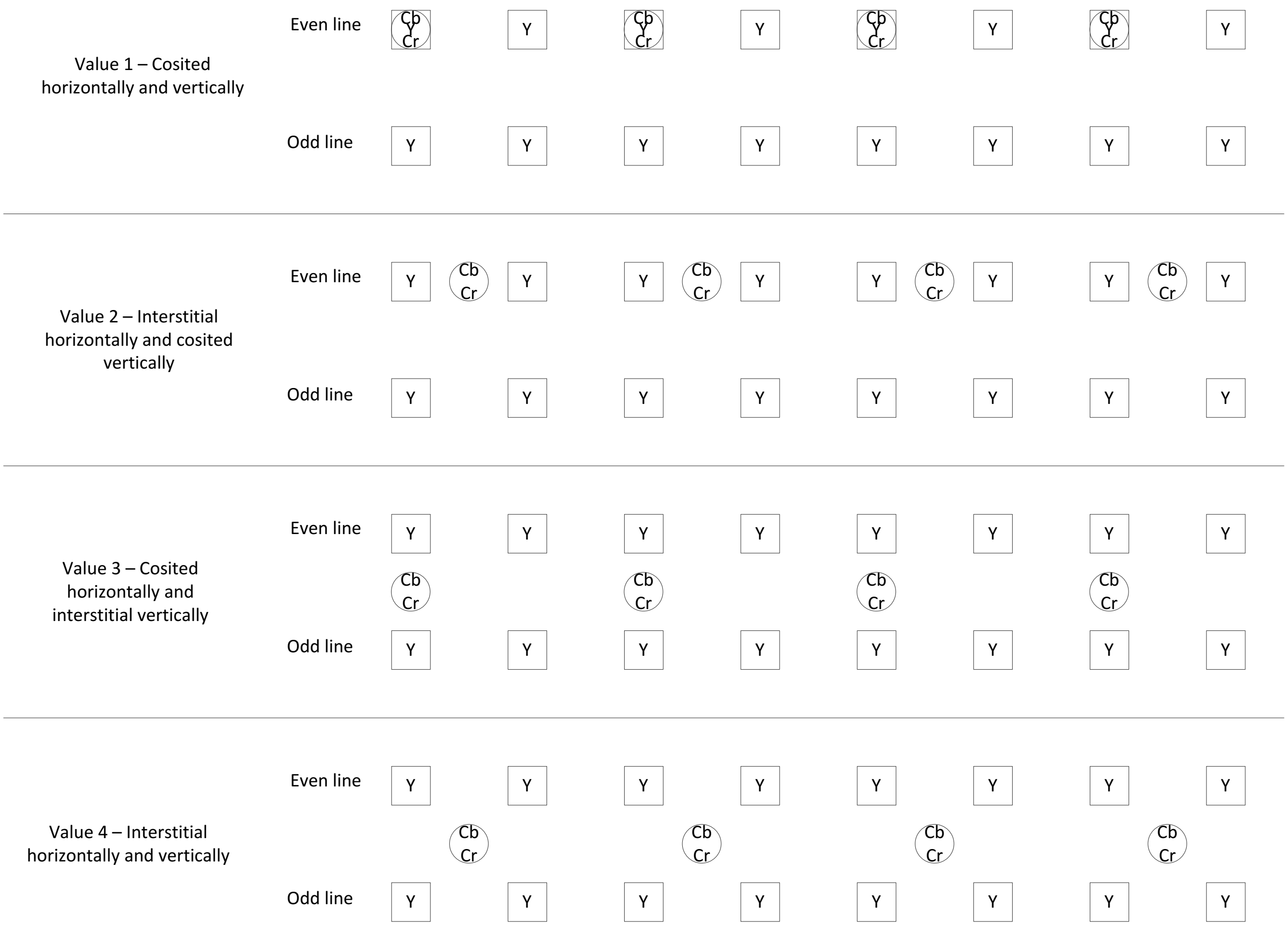

Figure 2 – Relative luminance and color difference sample positions for values of color-difference subsampling siting bits for 4:2:0 image elements

7.6 SMPTE Time Code and User Bits

Fields 58 and 59 (SMPTE time code and SMPTE user bits, both specified as U32) may be used to support time codes formatted according to either ST 12-1 or 12-3, which shall be indicated by the SMPTE time code type value in field 74.2 (which is defined in Table 15). Value 0 for field 74.2 is intended to provide a mechanism to allow conversion of legacy DPX files where the time code interpretation is uncertain; value 0 should not be used if the time code interpretation is known. Table 16 shows the mapping of the supported time code formats to the bits in fields 58 and 59.

Table 15 (field 74.2) – SMPTE time code type

Value	Description
0	Fields 58 and 59 shall be interpreted according to ST 268-1:2014 Table 6
1	Fields 58 and 59 shall be interpreted using the “ST 12-1 LTC bits” column of Table 16
2	Fields 58 and 59 shall be interpreted using the “ST 12-1 VITC bits” column of Table 16
3	Fields 58 and 59 shall be interpreted using the “ST 12-3 codeword bits” column of Table 16
4-254	Reserved

Note: The only difference in interpretation for values 1 and 2 (ST 12-1 LTC and ST 12-1 VITC) is the interpretation of the Modulation method specific flag which is used for Polarity correction in the LTC mapping and Field identification flag in the VITC mapping.

Table 16 – Mapping of fields 58 and 59 to time code formats

Field	Bits	Description (informative)	ST 12-1 LTC bits (as defined in section 9 of SMPTE ST 12-1:2014)	ST 12-1 VITC bits (as defined in section 10 of SMPTE ST 12-1:2014)	ST 12-3 codeword bits (as defined in section 7 of SMPTE 12-3:2016)
58	0 - 3	Units of frames	0 – 3	2 - 5	0 – 3
	4 – 7	Tens of frames and flags	8 – 11	12 - 15	8 - 11
	8 - 11	Units of seconds	16 – 19	22 - 25	16 - 19
	12 - 15	Tens of seconds and flags	24 – 27	32 - 35	24 - 27
	16 - 19	Units of minutes	32 – 35	42 - 45	32 - 35
	20 - 23	Tens of minutes and flags	40 – 43	52 - 55	40 - 43
	24 - 27	Units of hours	48 – 51	62 - 65	48 - 51
	28 - 31	Tens of hours	56 – 59	72 - 75	56 - 59
59	0 - 3	First binary group	4 – 7	6 - 9	4 - 7
	4 - 7	Second binary group	12 – 15	16 - 19	12 - 15
	8 - 11	Third binary group	20 – 23	26 - 29	20 - 23
	12 - 15	Fourth binary group	28 – 31	36 - 39	28 - 31
	16 - 19	Fifth binary group	36 – 39	46 - 49	36 - 39
	20 - 23	Sixth binary group	44 – 47	56 - 59	44 - 47
	24 - 27	Seventh binary group	52 – 55	66 - 69	52 - 55
	28 - 31	Eighth binary group	60 – 63	76 - 79	60 - 63

7.7 Video Signal Standard Extensions

Table 17 shall define video signal standard extensions.

Table 17 – Video signal standard extensions

Value	Signal Standard
250	Digital video described by Video Identification Code (VIC) in field 74.1

7.8 Standards-based Metadata Format Descriptor

Table 18 shall define the standards-based metadata format descriptor field.

Table 18 (field 85) – Standards-based metadata format descriptor

Value	Description
"ST336"	KLV (Key-Length-Value) metadata (as specified in ST 336)
"Reg-XML"	Reg-XML metadata (as specified in ST 2001-1)
"XMP"	XMP (Extensible Metadata Platform) metadata (as specified in ISO 16684-1) using UTF-8 character encoding (as specified in ISO/IEC 10646). Annex C provides an example of the XMP metadata format.

8 Component Data Packing Methods

8.1 General provisions related to component data packing

Datum values (as described in section 7.2.2) shall be mapped to 32-bit image data words using the packing method indicated by field 2x.10 and using the datum mapping direction specified in field 16.2. Annex B provides illustrations of some of these methods.

For a most-significant-byte-first file, image data words shall be in most-significant-byte-first order. For a least-significant-byte-first file, image data words shall be in least-significant-byte-first-order (i.e., using the opposite byte order as a most-significant-byte-first file).

Each raster line of image data shall be represented by an integer number of 32-bit image data words. All unused bits in the final 32-bit image data word of a raster line shall be zero. The padding specified by fields 2x.13 and 2x.14 shall be present after the final image data word of the line and image, respectively. The first datum value of each raster line shall be in the image data word that follows the padding (if present).

8.2 Packed into 32-bit words

When the value of field 2x.10 is equal to 0, datum values shall be packed adjacently with no padding bits in between, except at the ends of raster lines as described in section 8.1. Some examples are provided in Figure B.2, Figure B.3, and Figure B.8 through Figure B.15.

When the datum mapping direction (field 16.2) is equal to 0, datum values shall be packed into image data words in sequential order starting with the first datum value placed in the least-significant bit(s) of the image data word. Once an image data word is full, the packing shall proceed to the next image data word where no additional fill bits or zeroes shall be inserted until the end of the line. The following equations shall apply:

$$i_n = ((n * d + s_n * p) >> 5) + s_n * e / 4, \quad (1)$$

$$a_n = (n * d + s_n * p) \,\&\, 0x1f, \quad (2)$$

$$j_n = ((n * d + s_n * p + d - 1) >> 5) + s_n * e / 4, \quad (3)$$

$$b_n = (n * d + s_n * p + d - 1) \,\&\, 0x1f, \quad (4)$$

where

n	is the index of the datum within the image element (where n = 0 corresponds with the first datum),
d	is the bit depth of the datum values,
e	is the end-of-line padding (field 2x.13),
s_n	is the index of the raster line corresponding to datum n (where s_n = 0 corresponds with the first raster line) and is equal to $\lfloor n/q \rfloor$,
p	is equal to the number of zero-bits added to the final image data word at the end of each raster line,
q	is equal to the number of datum values in each raster line,
i_n	is the index of image data word (where i = 0 corresponds with the first image data word) where the least-significant bit of datum n is located,
a_n	is the bit position within the image data word where the least-significant bit of datum n is located,
j_n	is the index of image data word (where j = 0 corresponds with the first image data word) where the most-significant bit of datum n is located,
b_n	is the bit position within the image data word where the most-significant bit of datum n is located.

An example how these formulae map datum values to image data words is shown in Figure 3.

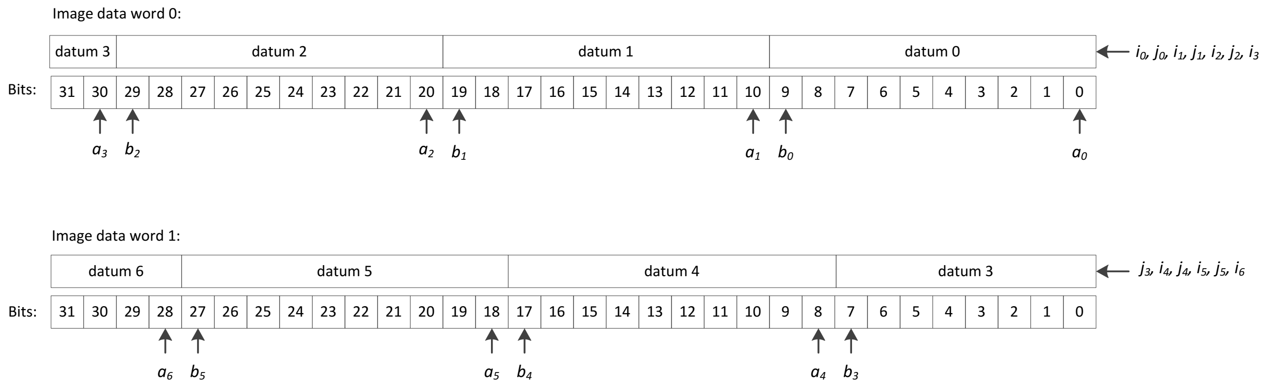

Figure 3 – Index and bit positions for first two image data words when *d* = 10 using datum mapping direction = 0

When datum mapping direction (field 16.2) is equal to 1, datum values shall be packed into image data words in sequential order starting with the first datum value placed in the most-significant bit(s) of the image data word. The following equations shall apply:

$$i_n = (((n + 1) * d + s_n * p - 1) >> 5) + s_n * e / 4, \quad (5)$$

$$a_n = (-(n + 1) * d - s_n * p) \,\&\, 0x1f, \quad (6)$$

$$j_n = ((n * d + s_n * p) >> 5) + s_n * e / 4, \quad (7)$$

$$b_n = (-n * d - s_n * p - 1) \,\&\, 0x1f, \quad (8)$$

where the variables are defined the same as in equations (1) through (4).

8.3 Filled to 32-bit words, method A

When the bit depth (field 2x.9) is equal to 10 or 12, a value of 1 for component data packing method may be used. The following provisions apply when the component data packing method (field 2x.10) is 1. Some examples are provided in Figure B.4 and Figure B.5.

When the datum mapping direction (field 16.2) is 0, unused image data word bits shall be zero, and the following equations shall apply:

$$i_n = \lfloor m/z \rfloor + s_n * e / 4, \quad (9)$$

$$a_n = \begin{cases} d * (m\%z) + r & if\ d \le 10 \\ 16 * (m\%z) + 16 - d & if\ d > 10 \end{cases} \quad (10)$$

$$j_n = \lfloor m/z \rfloor + s_n * e / 4, \quad (11)$$

$$b_n = \begin{cases} d * (m \% z) + r + d - 1 & if\ d \leq 10 \\ 16 * (m\%z) + 15 & if\ d > 10 \end{cases} \qquad (12)$$

where the variables are as defined in equations (1) through (4), and

m is the datum index with end-of-line datum padding and is equal to $n + (z * \lceil q/z \rceil - q) * s_n$,

z is the number of datum values per image data word and is equal to $\lfloor 32/d \rfloor$,

r is the number of zero-padding bits in each image data word and is equal to $32 - z * d$.

When the datum mapping direction (field 16.2) is 1, unused image data word bits shall be zero, and the following equations shall apply:

$$i_n = \lfloor m/z \rfloor + s_n * e / 4, \qquad (13)$$

$$a_n = \begin{cases} d * (z - 1 - (m \% z)) + r & if\ d \leq 10 \\ 32 - 16 * (m\%z) - d & if\ d > 10 \end{cases} \qquad (14)$$

$$j_n = \lfloor m/z \rfloor + s_n * e / 4, \qquad (15)$$

$$b_n = \begin{cases} d * (z - 1 - (m \% z)) + r + d - 1 & if\ d \leq 10 \\ 31 - 16 * (m\%z) & if\ d > 10 \end{cases} \qquad (16)$$

where the variables are as defined in equations (9) through (12).

8.4 Filled to 32-bit words, method B

When the bit depth (field 2x.9) is equal to 10 or 12, a value of 2 for component data packing method may be used. The following provisions apply when the component data packing method (field 2x.10) is 2. Some examples are provided in Figure B.6 and Figure B.7.

When the datum mapping direction (field 16.2) is 0, unused image data word bits shall be zero, and the following equations shall apply:

$$i_n = \lfloor m/z \rfloor + s_n * e / 4, \qquad (17)$$

$$a_n = \begin{cases} d * (m \% z) & if\ d \leq 10 \\ 16 * (m\%z) & if\ d > 10 \end{cases} \qquad (18)$$

$$j_n = \lfloor m/z \rfloor + s_n * e / 4, \qquad (19)$$

$$b_n = \begin{cases} d \ * \ (m \,\%\, z) + d - 1 & if\ d \leq 10 \\ 16 * (m\%z) + d - 1 & if\ d > 10 \end{cases} \tag{20}$$

where the variables are as defined in equations (9) through (12), and

When the datum mapping direction (field 16.2) is 1, unused image data word bits shall be zero, and the following equations shall apply:

$$i_n = \lfloor m/z \rfloor + s_n * e / 4, \tag{21}$$

$$a_n = \begin{cases} d \ * \ (z - 1 - (m \,\%\, z)) & if\ d \leq 10 \\ 16 - 16 * (m\%z) & if\ d > 10 \end{cases} \tag{22}$$

$$j_n = \lfloor m/z \rfloor + s_n * e / 4, \tag{23}$$

$$b_n = \begin{cases} d \ * \ (z - 1 - (m \,\%\, z)) + d - 1 & if\ d \leq 10 \\ 15 - 16 * (m\%z) + d & if\ d > 10 \end{cases} \tag{24}$$

where the variables are as defined in equations (9) through (12).

Annex A Byte Ordering (normative)

The order that the magic number (field 1) is stored in the file is either "SDPX" or "XPDS". If field 1 is ordered as "SDPX" in the file, the order of bytes within each field in the file shall be most-significant-byte-first. For ASCII fields, the bytes shall be stored in sequential order starting at the file offset specified in the field specification. U8 fields are single bytes and shall be stored at the file offset specified in the field specification. U16 fields are two bytes long and shall be stored at the file offset specified in the field specification, where the first byte shall contain the most-significant data byte and the second byte shall contain the least-significant data byte. R32 and U32 fields are four bytes long and shall be stored at the file offset specified in the field specification, where the first byte shall contain the most-significant data byte, the second byte shall contain the second-most-significant data byte, the third byte shall contain the second-least-significant data byte, and the fourth byte shall contain the least-significant data byte.

If field 1 is ordered as "XPDS" in the file, the order of bytes within each field in the file shall be least-significant-byte-first. For ASCII fields, the bytes shall be stored in sequential order starting at the file offset specified in the field specification. U8 fields are single bytes and shall be stored at the file offset specified in the field specification. U16 fields are two bytes long and shall be stored at the file offset specified in the field specification, where the first byte shall contain the least-significant data byte and the second byte shall contain the most-significant data byte. R32 and U32 fields are four bytes long and shall be stored at the file offset specified in the field specification, where the first byte shall contain the least-significant data byte, the second byte shall contain the second-least-significant data byte, the third byte shall contain the second-most-significant data byte, and the fourth byte shall contain the most-significant data byte.

Note: image data words (fields 77-84) are specified as U32 and follow the byte-order used for U32 fields.

Annex B Datum Packing Figures (informative)

B.1 General

The figures in this Annex illustrate the packing of different size image components into 32-bit storage words using most-significant-byte-first order.

In all of these figures, the bytes are shown in sequential order as they would exist when the byte order of the contents of field 1 in the DPX file is “`SDPX`” (i.e., most-significant-byte-first or “big-endian” order). Files that are stored in least-significant-byte-first order reverse the byte ordering of each 32-bit image data word (i.e., the order of the first and fourth bytes of the 32-bit word are swapped and the order of the second and third bytes of the 32-bit word are swapped). The figures do not illustrate the latter ordering (“little-endian”).

For each bit depth, datum values are packed in the manner shown in the figures based on the packing value (field 2x.10) for the image element and in the direction indicated by the datum mapping direction value (field 16.2).

Figure B.2 through Figure B.7 show examples of the first image data word in the image data section for an image element. Figure B.8 through Figure B.15 show examples of the first few image data words in the image data section for an image element.

Although not shown in the figures, the final image data word of each raster line is padded with zero bits if necessary so that the subsequent raster line starts on a 32-bit boundary. Additional padding using image data words is indicated by the end-of-line padding (field 2x.13) and end of image padding (field 2x.14).

<table>
<tr><td>Bytes in file order:</td><td colspan="8">Byte n</td><td colspan="8">Byte n+1</td><td colspan="8">Byte n+2</td><td colspan="8">Byte n+3</td></tr>
<tr><td>8-bit component(s):</td><td colspan="8">d3</td><td colspan="8">d2</td><td colspan="8">d1</td><td colspan="8">d0</td></tr>
<tr><td>Bits of each component:</td><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr>
</table>

Figure B.1 – Example illustrating the nomenclature used in figures

Figure B.1 is a representative example to illustrate the nomenclature that is used in the figures that follow. The “bytes in file order” line illustrates bytes in sequential order as they would appear in a most-significant-byte-first (big-endian) file, and the bits comprising each such byte are shown from most-significant to least-significant from left to right. “Byte n” refers to the location of the first image data word in the image data section for an image element. Image data words are required to be aligned to a 32-bit boundary with respect to the start of the file; therefore, the offset for “byte n” is a multiple of 4.

The second “component(s)” line illustrates the location of datum values d0, d1, d2, etc., within the image data words. The datum values are component-wise image samples in the component order specified by the image element descriptor (field 2x.6) and in the scan order indicated by the image orientation (field 17). Figure B.8 through Figure B.11 show some specific cases when the image element descriptor (field 2x.6) indicates the samples are RGB (R0, G0, B0 is the first pixel in the image scan order; R1, G1, B1 is the second pixel; etc.).

The “bits of each component” row shows the location of individual bits of each datum value. The bits are labeled b0, b1, b2, etc., where b0 is the least-significant bit of the datum value.

Table B.1 – List of datum packing figures

Figure	Bit depth(s)	Packing (field 2x.10)	Datum mapping direction (field 16.2)	Single or multiple image data words
Figure B.2	1, 8, 10, 12, 16	packed (0)	right-to-left (0)	Single
Figure B.3	1, 8, 10, 12, 16	packed (0)	left-to-right (1)	Single
Figure B.4	10, 12	filled method A (1)	right-to-left (0)	Single
Figure B.5	10, 12	filled method A (1)	left-to-right (1)	Single
Figure B.6	10, 12	filled method B (2)	right-to-left (0)	Single
Figure B.7	10, 12	filled method B (2)	left-to-right (1)	Single
Figure B.8	8 (RGB)	packed (0)	right-to-left (0)	Multiple
Figure B.9	8 (RGB)	packed (0)	left-to-right (1)	Multiple
Figure B.10	8 (BGR)	packed (0)	right-to-left (0)	Multiple
Figure B.11	8 (BGR)	packed (0)	left-to-right (1)	Multiple
Figure B.12	10	packed (0)	right-to-left (0)	Multiple
Figure B.13	10	packed (0)	left-to-right (1)	Multiple
Figure B.14	12	packed (0)	right-to-left (0)	Multiple
Figure B.15	12	packed (0)	left-to-right (1)	Multiple

Bytes in file order:	Byte n								Byte n+1								Byte n+2								Byte n+3							
1-bit component(s):	d31	d30	d29	d28	d27	d26	d25	d24	d23	d22	d21	d20	d19	d18	d17	d16	d15	d14	d13	d12	d11	d10	d9	d8	d7	d6	d5	d4	d3	d2	d1	d0

Bytes in file order:	Byte n								Byte n+1								Byte n+2								Byte n+3							
8-bit component(s):	d3								d2								d1								d0							
Bits of each component:	b7	b6	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0

Bytes in file order:	Byte n								Byte n+1								Byte n+2								Byte n+3							
10-bit component(s):	d3		d2										d1										d0									
Bits of each component:	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0

Bytes in file order:	Byte n								Byte n+1								Byte n+2								Byte n+3							
12-bit component(s):	d2								d1												d0											
Bits of each component:	b7	b6	b5	b4	b3	b2	b1	b0	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0

Bytes in file order:	Byte n								Byte n+1								Byte n+2								Byte n+3							
16-bit component(s):	d1																d0															
Bits of each component:	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0

Figure B.2 – Image data word format for packed packing (field 2x.10 = 0) using right-to-left datum mapping direction (field 16.2 = 0)

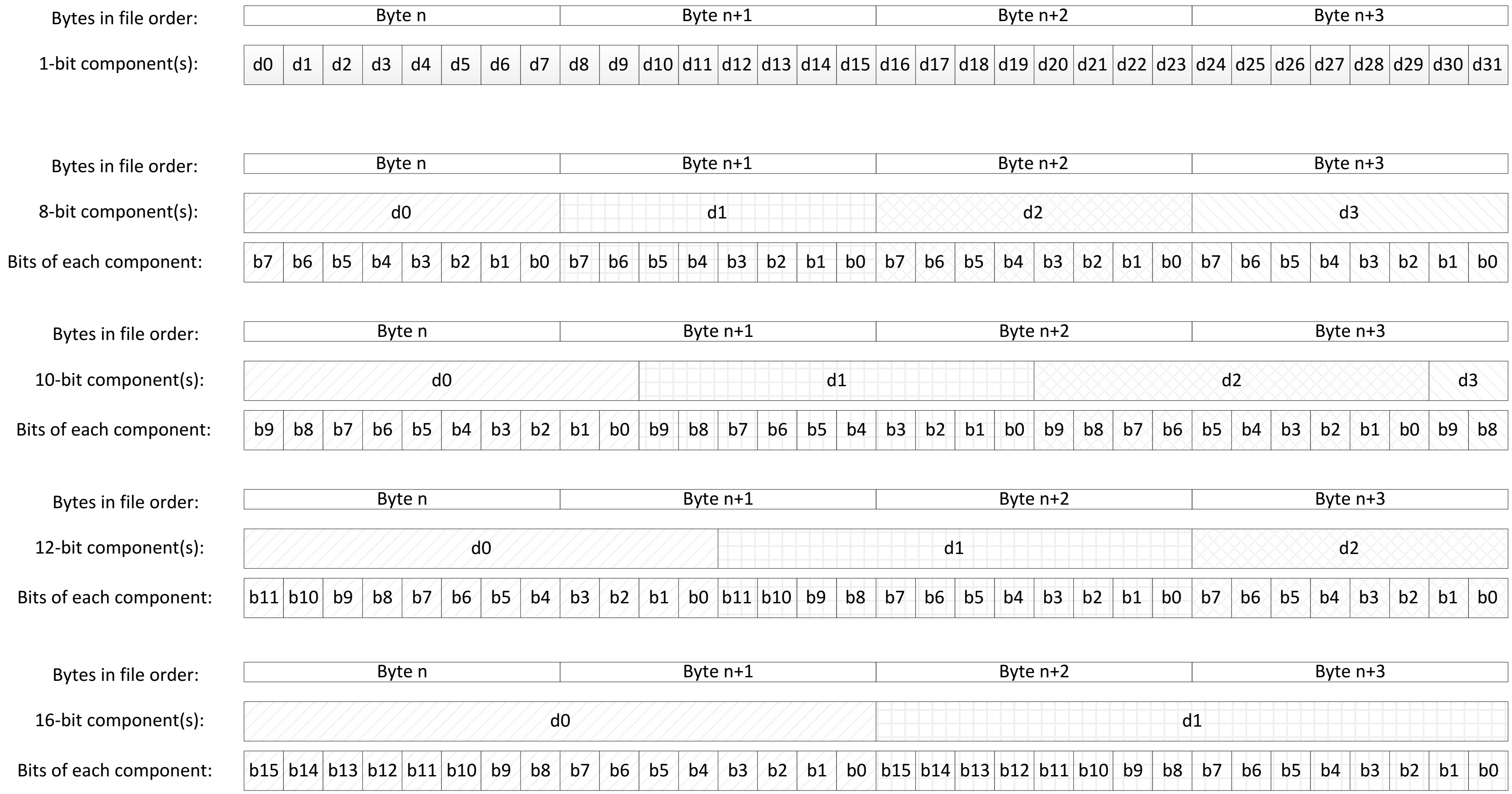

Figure B.3 – Image data word format for packed packing (field 2x.10 = 0) using left-to-right datum mapping direction (field 16.2 = 1)

Bytes in file order:	Byte n								Byte n+1								Byte n+2								Byte n+3							
10-bit component(s):	d2										d1										d0										0	0
Bits of each component:	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	0	0

Bytes in file order:	Byte n								Byte n+1								Byte n+2								Byte n+3							
12-bit component(s):	d1												0	0	0	0	d0												0	0	0	0
Bits of each component:	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	0	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	0

Figure B.4 – Image data word format for filled method A packing (field 2x.10 = 1) using right-to-left datum mapping direction (field 16.2 = 0)

Bytes in file order:	Byte n								Byte n+1								Byte n+2								Byte n+3							
10-bit component(s):	d0										d1										d2										0	0
Bits of each component:	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	0	0

Bytes in file order:	Byte n								Byte n+1								Byte n+2								Byte n+3							
12-bit component(s):	d0												0	0	0	0	d1												0	0	0	0
Bits of each component:	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	0	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	0

Figure B.5 – Image data word format for filled method A packing (field 2x.10 = 1) using left-to-right datum mapping direction (field 16.2 = 1)

Bytes in file order:	Byte n								Byte n+1								Byte n+2								Byte n+3							
10-bit component(s):	0	0	d2										d1										d0									
Bits of each component:	0	0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0

Bytes in file order:	Byte n								Byte n+1								Byte n+2								Byte n+3							
12-bit component(s):	0	0	0	0	d1												0	0	0	0	d0											
Bits of each component:	0	0	0	0	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	0	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0

Figure B.6 – Image data word format for filled method B packing (field 2x.10 = 2) using right-to-left datum mapping direction (field 16.2 = 0)

Bytes in file order:	Byte n								Byte n+1								Byte n+2								Byte n+3							
10-bit component(s):	0	0	d0										d1										d2									
Bits of each component:	0	0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0

Bytes in file order:	Byte n								Byte n+1								Byte n+2								Byte n+3							
12-bit component(s):	0	0	0	0	d0												0	0	0	0	d1											
Bits of each component:	0	0	0	0	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	0	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0

Figure B.7 – Image data word format for filled method B packing (field 2x.10 = 2) using left-to-right datum mapping direction (field 16.2 = 1)

Bytes in file order:	Byte n	Byte n+1	Byte n+2	Byte n+3
8-bit component(s):	R1	B0	G0	R0
Bytes in file order:	Byte n+4	Byte n+5	Byte n+6	Byte n+7
8-bit component(s):	G2	R2	B1	G1
Bytes in file order:	Byte n+8	Byte n+9	Byte n+10	Byte n+11
8-bit component(s):	B3	G3	R3	B2

Figure B.8 – 8-bit RGB component (field 2x.6 = 56) packing using right-to-left datum mapping direction (field 16.2 = 0)

Bytes in file order:	Byte n	Byte n+1	Byte n+2	Byte n+3
8-bit component(s):	R0	G0	B0	R1
Bytes in file order:	Byte n+4	Byte n+5	Byte n+6	Byte n+7
8-bit component(s):	G1	B1	R2	G2
Bytes in file order:	Byte n+8	Byte n+9	Byte n+10	Byte n+11
8-bit component(s):	B2	R3	G3	B3

Figure B.9 – 8-bit RGB component (field 2x.6 = 56) packing using left-to-right datum mapping direction (field 16.2 = 1)

Bytes in file order:	Byte n	Byte n+1	Byte n+2	Byte n+3
8-bit component(s):	B1	R0	G0	B0
Bytes in file order:	Byte n+4	Byte n+5	Byte n+6	Byte n+7
8-bit component(s):	G2	B2	R1	G1
Bytes in file order:	Byte n+8	Byte n+9	Byte n+10	Byte n+11
8-bit component(s):	R3	G3	B3	R2

Figure B.10 – 8-bit BGR component (field 2x.6 = 53) packing using right-to-left datum mapping direction (field 16.2 = 0)

Bytes in file order:	Byte n	Byte n+1	Byte n+2	Byte n+3
8-bit component(s):	B0	G0	R0	B1
Bytes in file order:	Byte n+4	Byte n+5	Byte n+6	Byte n+7
8-bit component(s):	G1	R1	B2	G2
Bytes in file order:	Byte n+8	Byte n+9	Byte n+10	Byte n+11
8-bit component(s):	R2	B3	G3	R3

Figure B.11 – 8-bit BGR component (field 2x.6 = 53) packing using left-to-right datum mapping direction (field 16.2 = 1)

Bytes in file order:	Byte n								Byte n+1								Byte n+2								Byte n+3							
10-bit component(s):	d3		d2										d1										d0									
Bits of each component:	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0

Bytes in file order:	Byte n+4								Byte n+5								Byte n+6								Byte n+7							
10-bit component(s):	d6				d5										d4										d3							
Bits of each component:	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2

Bytes in file order:	Byte n+8								Byte n+9								Byte n+10								Byte n+11							
10-bit component(s):	d9						d8										d7										d6					
Bits of each component:	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4

Bytes in file order:	Byte n+12								Byte n+13								Byte n+14								Byte n+15							
10-bit component(s):	d12								d11										d10										d9			
Bits of each component:	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6

Bytes in file order:	Byte n+16								Byte n+17								Byte n+18								Byte n+19							
10-bit component(s):	d15										d14										d13										d12	
Bits of each component:	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8

Figure B.12 – 10-bit component packing (field 2x.10 = 0) using right-to-left datum mapping direction (field 16.2 = 0)

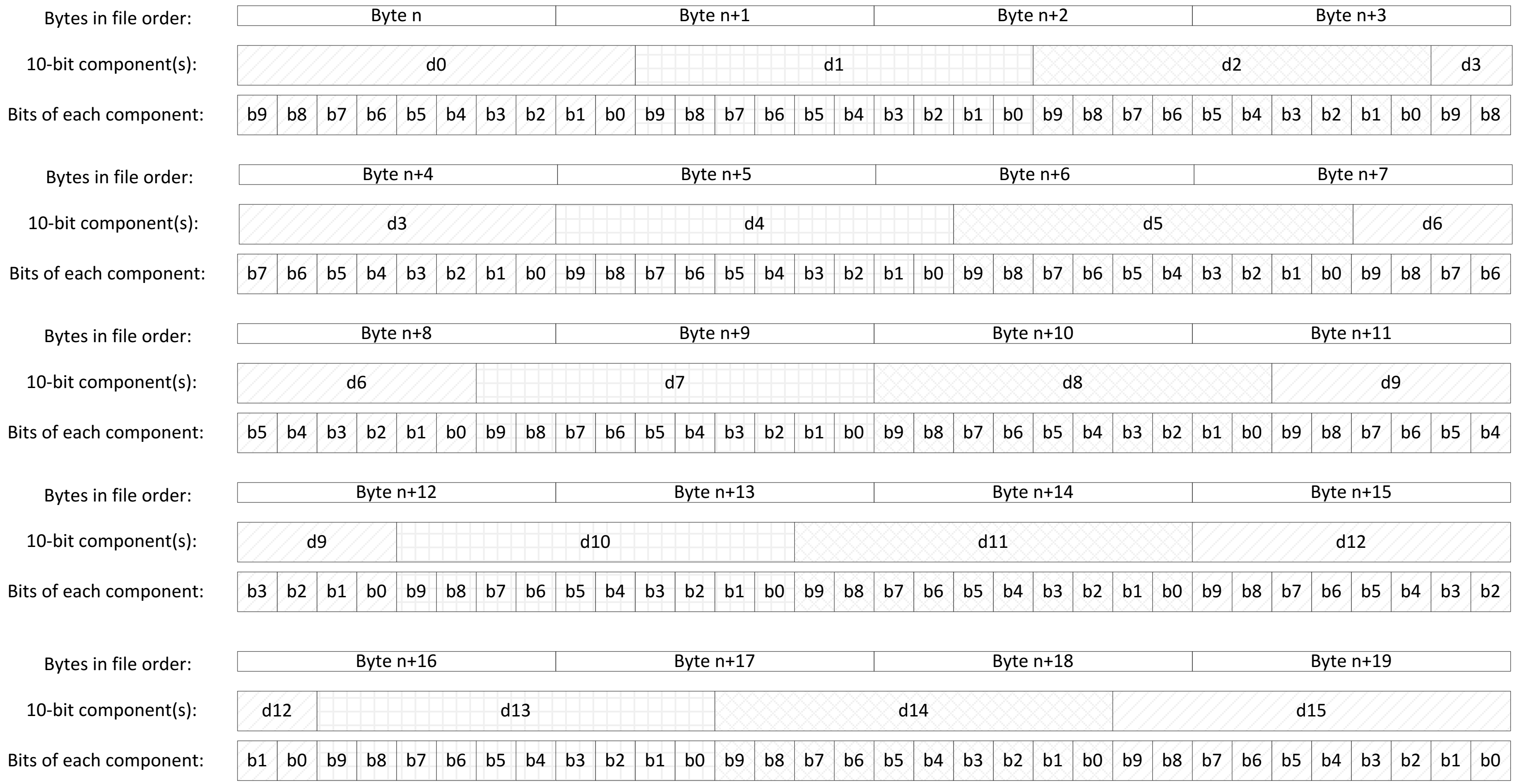

Figure B.13 – 10-bit component packing (field 2x.10 = 0) using left-to-right datum mapping direction (field 16.2 = 1)

<table>
<tr><td>Bytes in file order:</td><td colspan="8">Byte n</td><td colspan="8">Byte n+1</td><td colspan="8">Byte n+2</td><td colspan="8">Byte n+3</td></tr>
<tr><td>12-bit component(s):</td><td colspan="8">d2</td><td colspan="12">d1</td><td colspan="12">d0</td></tr>
<tr><td>Bits of each component:</td><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td><td>b11</td><td>b10</td><td>b9</td><td>b8</td><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td><td>b11</td><td>b10</td><td>b9</td><td>b8</td><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr>
</table>

<table>
<tr><td>Bytes in file order:</td><td colspan="8">Byte n+4</td><td colspan="8">Byte n+5</td><td colspan="8">Byte n+6</td><td colspan="8">Byte n+7</td></tr>
<tr><td>12-bit component(s):</td><td colspan="4">d5</td><td colspan="12">d4</td><td colspan="12">d3</td><td colspan="4">d2</td></tr>
<tr><td>Bits of each component:</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td><td>b11</td><td>b10</td><td>b9</td><td>b8</td><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td><td>b11</td><td>b10</td><td>b9</td><td>b8</td><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td><td>b11</td><td>b10</td><td>b9</td><td>b8</td></tr>
</table>

<table>
<tr><td>Bytes in file order:</td><td colspan="8">Byte n+8</td><td colspan="8">Byte n+9</td><td colspan="8">Byte n+10</td><td colspan="8">Byte n+11</td></tr>
<tr><td>12-bit component(s):</td><td colspan="12">d7</td><td colspan="12">d6</td><td colspan="8">d5</td></tr>
<tr><td>Bits of each component:</td><td>b11</td><td>b10</td><td>b9</td><td>b8</td><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td><td>b11</td><td>b10</td><td>b9</td><td>b8</td><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td><td>b11</td><td>b10</td><td>b9</td><td>b8</td><td>b7</td><td>b6</td><td>b5</td><td>b4</td></tr>
</table>

Figure B.14 – 12-bit component packing (field 2x.10 = 0) using right-to-left datum mapping direction (field 16.2 = 0)

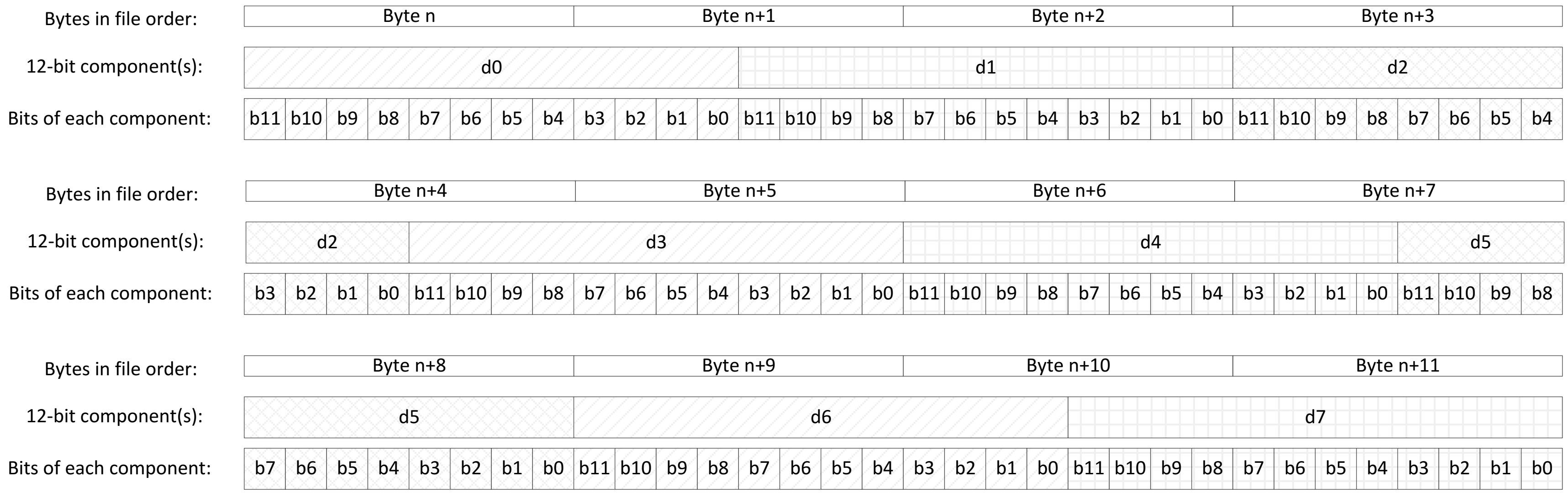

Figure B.15 – 12-bit component packing (field 2x.10 = 0) using left-to-right datum mapping direction (field 16.2 = 1)

B.2 Run-length encoding

When the component data encoding (field 2x.11) is set to 1, the image element pixel data is encoded using run-length encoding. In this case, the first datum comprises a count and a flag. The *count* is contained within the most-significant bits and the *flag* is contained within the least-significant bit. If the flag is equal to 0, the next *count* pixels are encoded sequentially using the component order indicated by the image element descriptor (field 2x.6) in the following datum values. If the flag is 1, the next *count* pixels are all the same pixel value, which is indicated by the next datum values in the component order indicated by the image element descriptor (field 2x.6).

Figure B.16 shows an example of run-length encoding where pixels 2 through 6 are all the same value.

Bytes in file order:	Byte n								Byte n+1								Byte n+2								Byte n+3							
10-bit component(s):	B0		G0										R0										0x2									0
Bits of each component:	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0

Bytes in file order:	Byte n+4								Byte n+5								Byte n+6								Byte n+7							
10-bit component(s):	B1				G1										R1										B0							
Bits of each component:	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2

Bytes in file order:	Byte n+8								Byte n+9								Byte n+10								Byte n+11							
10-bit component(s):	G2						R2										0x5									1	B1					
Bits of each component:	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4

Bytes in file order:	Byte n+12								Byte n+13								Byte n+14								Byte n+15							
10-bit component(s):	R7								0x2									0	B2										G2			
Bits of each component:	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6

Bytes in file order:	Byte n+16								Byte n+17								Byte n+18								Byte n+19							
10-bit component(s):	R8										B7										G7										R7	
Bits of each component:	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b9	b8

Figure B.16 – Example of run-length encoded 10-bit RGB component packing (field 2x.6 = 56, field 2x.10 = 0) using right-to-left datum mapping direction (field 16.2 = 0)

Annex C Standards-Based Metadata (informative)

C.1 General

This Standard allows standards-based metadata to be carried in field 87. For HDR pictures, examples of useful metadata include ST 2086 static metadata or dynamic metadata as defined in ST 2094-2.

In cases where the metadata described in a field in the standards-based metadata has the same meaning as metadata that is present in the DPX header, it is generally recommended that the DPX header metadata takes precedence in order to maximize interoperability. A DPX header metadata item could be considered to be not present if it has an undefined value. For some types of textual metadata, a standards-based representation could potentially be more descriptive (e.g., if it allows Unicode characters that cannot be represented using ASCII strings).

C.2 Extensible Metadata Platform (XMP)

If XMP metadata is used, the standards-based metadata format descriptor (field 85) is equal to "`XMP`". The metadata representation is in an XML-based format as described in ISO 16684-1. Metadata items from the SMPTE Registry (http://smpte-ra.org) can be included.

An example of an XMP representation is provided below.

```
<?xpacket begin="" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="Adobe XMP Core 5.6-c138 79.159824, 2016/09/14-
01:09:01        ">
   <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
      <rdf:Description rdf:about=""
            xmlns:xmp="http://ns.adobe.com/xap/1.0/"
            xmlns:dc="http://purl.org/dc/elements/1.1/"
            xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/"
            xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"
            xmlns:stEvt="http://ns.adobe.com/xap/1.0/sType/ResourceEvent#"
            xmlns:tiff="http://ns.adobe.com/tiff/1.0/"
            xmlns:exif="http://ns.adobe.com/exif/1.0/"
            xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/"
            xmlns:xmpDM="http://ns.adobe.com/xmp/1.0/DynamicMedia/"
            xmlns:plus="http://ns.useplus.org/ldf/xmp/1.0/"
            xmlns:Iptc4xmpExt="http://iptc.org/std/Iptc4xmpExt/2008-02-29/"
            xmlns:r0="http://www.smpte-ra.org/reg/395/2014/13/1/aaf"
            xmlns:r1="http://www.smpte-ra.org/reg/335/2012/"
            xmlns:r2="http://www.smpte-ra.org/reg/2003/2012/">
         <xmp:CreatorTool>Adobe Photoshop CC 2017 (Macintosh)</xmp:CreatorTool>
         <xmp:CreateDate>2017-02-21T12:50:54-10:00</xmp:CreateDate>
         <xmp:ModifyDate>2017-02-21T15:44:23-10:00</xmp:ModifyDate>
         <xmp:MetadataDate>2017-02-21T15:44:23-10:00</xmp:MetadataDate>
         <dc:format>application/vnd.adobe.photoshop</dc:format>
         <dc:creator>
            <rdf:Seq>
```

```
               <rdf:li>John Doe</rdf:li>
            </rdf:Seq>
         </dc:creator>
         <dc:rights>
            <rdf:Alt>
               <rdf:li xml:lang="x-default">2017 John Doe</rdf:li>
            </rdf:Alt>
         </dc:rights>
         <dc:title>
            <rdf:Alt>
               <rdf:li xml:lang="x-default">XMP metadata example</rdf:li>
            </rdf:Alt>
         </dc:title>
         <photoshop:ColorMode>3</photoshop:ColorMode>
         <photoshop:ICCProfile>HDR Rec.2100 PQ - SDR white 200 nits</photoshop:ICCProfile>
         <photoshop:DateCreated>2016-12-15T11:44:46</photoshop:DateCreated>
         <photoshop:AuthorsPosition>Standardizer</photoshop:AuthorsPosition>
         <photoshop:Headline>Fake News</photoshop:Headline>
         <xmpMM:InstanceID>xmp.iid:d4d21749-edaf-4d5a-91dc-0255c5a8fcfe</xmpMM:InstanceID>
         <xmpMM:DocumentID>adobe:docid:photoshop:cc9b94eb-392e-117a-8042-
92210eed22ac</xmpMM:DocumentID>
         <xmpMM:OriginalDocumentID>xmp.did:8b8a8927-c6ad-448e-bd03-
5166b455eb40</xmpMM:OriginalDocumentID>
         <xmpMM:History>
            <rdf:Seq>
               <rdf:li rdf:parseType="Resource">
                  <stEvt:action>created</stEvt:action>
                  <stEvt:instanceID>xmp.iid:8b8a8927-c6ad-448e-bd03-
5166b455eb40</stEvt:instanceID>
                  <stEvt:when>2017-02-21T12:50:54-10:00</stEvt:when>
                  <stEvt:softwareAgent>Adobe Photoshop CC 2017
(Macintosh)</stEvt:softwareAgent>
               </rdf:li>
               <rdf:li rdf:parseType="Resource">
                  <stEvt:action>saved</stEvt:action>
                  <stEvt:instanceID>xmp.iid:d4d21749-edaf-4d5a-91dc-
0255c5a8fcfe</stEvt:instanceID>
                  <stEvt:when>2017-02-21T15:44:23-10:00</stEvt:when>
                  <stEvt:softwareAgent>Adobe Photoshop CC 2017
(Macintosh)</stEvt:softwareAgent>
                  <stEvt:changed>/</stEvt:changed>
               </rdf:li>
            </rdf:Seq>
         </xmpMM:History>
         <tiff:Orientation>1</tiff:Orientation>
         <tiff:XResolution>720000/10000</tiff:XResolution>
         <tiff:YResolution>720000/10000</tiff:YResolution>
         <tiff:ResolutionUnit>2</tiff:ResolutionUnit>
                  <tiff:ImageWidth>3888</tiff:ImageWidth>
         <tiff:ImageLength>2592</tiff:ImageLength>
         <tiff:BitsPerSample>
            <rdf:Seq>
               <rdf:li>8</rdf:li>
               <rdf:li>8</rdf:li>
               <rdf:li>8</rdf:li>
            </rdf:Seq>
         </tiff:BitsPerSample>
         <tiff:PhotometricInterpretation>2</tiff:PhotometricInterpretation>
```

```
        <tiff:SamplesPerPixel>3</tiff:SamplesPerPixel>
        <tiff:WhitePoint>
           <rdf:Seq>
              <rdf:li>313/1000</rdf:li>
              <rdf:li>329/1000</rdf:li>
           </rdf:Seq>
        </tiff:WhitePoint>
        <tiff:PrimaryChromaticities>
           <rdf:Seq>
              <rdf:li>64/100</rdf:li>
              <rdf:li>33/100</rdf:li>
              <rdf:li>21/100</rdf:li>
              <rdf:li>71/100</rdf:li>
              <rdf:li>15/100</rdf:li>
              <rdf:li>6/100</rdf:li>
           </rdf:Seq>
        </tiff:PrimaryChromaticities>
        <exif:ExifVersion>0221</exif:ExifVersion>
        <exif:FlashpixVersion>0100</exif:FlashpixVersion>
        <exif:DateTimeOriginal>2016-12-15T11:44:46</exif:DateTimeOriginal>
        <exif:ExposureTime>1/30</exif:ExposureTime>
        <exif:FNumber>36/1</exif:FNumber>
        <exif:ExposureProgram>4</exif:ExposureProgram>
        <exif:ISOSpeedRatings>
           <rdf:Seq>
              <rdf:li>1600</rdf:li>
           </rdf:Seq>
        </exif:ISOSpeedRatings>
        <exif:ShutterSpeedValue>321578/65536</exif:ShutterSpeedValue>
        <exif:ApertureValue>10339850/1000000</exif:ApertureValue>
        <exif:ExposureBiasValue>0/3</exif:ExposureBiasValue>
        <exif:MeteringMode>6</exif:MeteringMode>
        <exif:Flash rdf:parseType="Resource">
           <exif:Fired>False</exif:Fired>
           <exif:Return>0</exif:Return>
           <exif:Mode>2</exif:Mode>
           <exif:Function>False</exif:Function>
           <exif:RedEyeMode>False</exif:RedEyeMode>
        </exif:Flash>
        <exif:FocalLength>49/1</exif:FocalLength>
        <exif:FocalPlaneXResolution>3888000/877</exif:FocalPlaneXResolution>
        <exif:FocalPlaneYResolution>2592000/582</exif:FocalPlaneYResolution>
        <exif:FocalPlaneResolutionUnit>2</exif:FocalPlaneResolutionUnit>
        <exif:CustomRendered>0</exif:CustomRendered>
        <exif:ExposureMode>0</exif:ExposureMode>
        <exif:WhiteBalance>0</exif:WhiteBalance>
        <exif:SceneCaptureType>0</exif:SceneCaptureType>
        <exif:ColorSpace>1</exif:ColorSpace>
        <exif:PixelXDimension>1920</exif:PixelXDimension>
        <exif:PixelYDimension>1080</exif:PixelYDimension>
        <xmpRights:Marked>True</xmpRights:Marked>
        <xmpDM:tapeName>Reel 3</xmpDM:tapeName>
        <xmpDM:scene>27</xmpDM:scene>
        <xmpDM:shotName>2</xmpDM:shotName>
        <xmpDM:altTimecode rdf:parseType="Resource">
           <xmpDM:timeValue>01:00:00:23</xmpDM:timeValue>
        </xmpDM:altTimecode>
        <plus:ImageSupplier>
```

```
            <rdf:Seq>
               <rdf:li rdf:parseType="Resource"/>
            </rdf:Seq>
         </plus:ImageSupplier>
         <r0:RGBADescriptor>
            <r1:MasteringDisplayMaximumLuminance>
               100000000
            </r1:MasteringDisplayMaximumLuminance>
            <r1:MasteringDisplayMinimumLuminance>0</r1:MasteringDisplayMinimumLuminance>
            <r1:MasteringDisplayWhitePointChromaticity>
               <r2:X>15635</r2:X>
               <r2:Y>16450</r2:Y>
            </r1:MasteringDisplayWhitePointChromaticity>
            <r1:MasteringDisplayPrimaries>
               <r2:ColorPrimary>
                  <r2:X>34000</r2:X>
                  <r2:Y>16000</r2:Y>
               </r2:ColorPrimary>
               <r2:ColorPrimary>
                  <r2:X>13250</r2:X>
                  <r2:Y>34500</r2:Y>
               </r2:ColorPrimary>
               <r2:ColorPrimary>
                  <r2:X>7500</r2:X>
                  <r2:Y>3000</r2:Y>
               </r2:ColorPrimary>
            </r1:MasteringDisplayPrimaries>
         </r0:RGBADescriptor>
         <Iptc4xmpExt:RegistryId>
            <rdf:Bag>
               <rdf:li rdf:parseType="Resource"/>
            </rdf:Bag>
         </Iptc4xmpExt:RegistryId>

<Iptc4xmpExt:DigitalSourceType>http://cv.iptc.org/newscodes/digitalsourcetype/digitalCapture<
/Iptc4xmpExt:DigitalSourceType>
      </rdf:Description>
        <rdf:Description rdf:about=""
                    xmlns:aux="http://ns.adobe.com/exif/1.0/aux/">
                <aux:SerialNumber>520171488</aux:SerialNumber>
                <aux:LensInfo>18/1 250/1 0/0 0/0</aux:LensInfo>
                aux:Lens="EF24-70mm f/2.8L USM"
                aux:LensID="230"
                aux:LensSerialNumber="0000000000"
                aux:ImageNumber="0"
                aux:ApproximateFocusDistance="265/100"
                aux:FlashCompensation="0/1"
                aux:Firmware="1.2.3"
               </rdf:Description>
   </rdf:RDF>
</x:xmpmeta>
 <?xpacket end="w"?>
```

Bibliography (Informative)

SMPTE ST 2086:2014, Mastering Display Color Volume Metadata Supporting High Luminance and Wide Color Gamut Images

SMPTE ST 2094-2:2017, Dynamic Metadata for Color Volume Transform – KLV Encoding and MXF Mapping