SMPTE RP 2073-2:2017
Revision of SMPTE RP 2073-2:2015

SMPTE RECOMMENDED PRACTICE

VC-5 Video Essence – Part 2: Conformance Specification

Table of Contents

		Pages
Foreword		**3**
Intellectual Property		**3**
Introduction		**3**
1	**Scope**	**4**
2	**Conformance Notation**	**4**
3	**Normative References**	**5**
4	**Terms and Definitions**	**5**
5	**Overview (Informative)**	**5**
6	**Description of the Test Materials**	**6**
6.1	Source Code	6
6.2	VC-5 Parts	6
6.3	Reference Decoder	7
6.4	Sample Encoder	10
6.5	Image Converter	13
6.6	Image Comparer	14
6.7	Test Cases	15
6.8	Testing Layers	15
6.9	Testing Sections	15
6.10	Testing Image Sections	15
6.11	Testing Image Section Layers	16
7	**File Formats**	**16**
7.1	Image File Formats	16

Approved November 23, 2017

7.2 Unformatted Image Files 18
7.3 Component Array Files 18
7.4 DPX Image Files 18
7.5 Other Image File Formats 18
7.6 Bitstream File Formats 18
8 Conformance Testing (Informative) 19
8.1 Codec Test Script 19
8.2 Comprehensive Test Suite 21
9 Conformance Specification 22
9.1 Test Materials 22
9.2 Bitstream Conformance 22
9.3 Decoder Conformance 22
10 Installing and Building the Test Materials 23
10.1 Test Materials Installation 23
10.2 Test Materials Contents 23
10.3 Build Scripts and Documentation 24
10.4 Test Materials Media 24
10.5 Building the Encoder and Decoder 25
Annex A Reference Bitstreams (Normative) 26
A.1 Bitstream Files 26
A.2 VC-5 Part 1 Bitstreams 26
A.3 VC-5 Part 3 Bitstreams 28
A.4 VC-5 Part 4 Bitstreams 29
A.5 VC-5 Part 5 Bitstreams 30
A.6 VC-5 Part 6 Bitstreams 31
Annex B Reference Images (Normative) 34
B.1 Image Files 34
B.2 VC-5 Part 1 Images 34
B.3 VC-5 Part 3 Images 36
B.4 VC-5 Part 4 Images 37
B.5 VC-5 Part 5 Images 38
B.6 VC-5 Part 6 Images 40
Bibliography 43

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 10E.

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

This section is entirely informative and does not form an integral part of this Engineering Document.

VC-5 is a wavelet-based intra-frame codec intended for applications that require fast decoding and encoding with high visual quality. The VC-5 codec is suitable for image capture and post production.

This document describes the test materials and procedures for verifying that a bitstream produced by an implementation of a VC-5 encoder or an implementation of a VC-5 decoder is compliant with the VC-5 essence standard.

1 Scope

The VC-5 essence standard comprises the SMPTE standards designated SMPTE ST 2073-1, ST 2073-3, ST 2073-4, ST 2073-5, and ST 2073-6.

This recommended practice specifies the criteria and procedures for testing the conformance of encoder and decoder implementations to the VC-5 essence standard and describes the test materials used for conformance testing.

The test materials used for conformance testing comprise:

Reference decoder for testing bitstreams created by implementations of a VC-5 encoder for compliance with the VC-5 essence standard,

Bitstreams and reference images for testing implementations of a VC-5 decoder for compliance with the VC-5 essence standard.

Although not required for conformance testing, the test materials include a sample encoder, the images used by the sample encoder to create the bitstreams used for conformance testing, a program for converting between image formats, and a program for comparing image files. This recommended practice describes the image file formats used by the software and scripts that automate the conformance testing procedure.

Detailed instructions for installing and building the programs and contact information for submitting bug reports are provided in a separate document that is included in the software distribution of the test materials.

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.

SMPTE ST 2073-1:2017 VC-5 Video Essence. Part 1: Elementary Bitstream

SMPTE ST 2073-3:2015 VC-5 Video Essence. Part 3: Image Formats

SMPTE ST 2073-4:2015 VC-5 Video Essence. Part 4: Subsampled Color Difference Components

SMPTE VC-5 Test Materials, Version 4.3 or later

4 Terms and Definitions

For the purposes of this document, the following terms and definitions shall apply.

4.1 little-endian

order of bytes that comprise a number with less significant bytes before more significant bytes

4.2 format specification string

character string that specifies the format of the string output by the C language sprintf function

4.3 VC-5 essence standard

SMPTE standards designated SMPTE ST 2073-1, ST 2073-3, ST 2073-4, ST 2073-5 and ST 2073-6

5 Overview (Informative)

The test materials provided as part of this conformance specification include:

Source code for a reference decoder,

Source code for a sample encoder,

Utility programs and scripts useful for testing VC-5 decoder and encoder implementations,

Reference bitstreams for testing conformance of a decoder implementation to the VC-5 essence standard,

Images used to encode the reference bitstreams.

The bitstream defined in the VC-5 essence standard consists of an ordered set of component arrays. In typical practice, each component array corresponds to an image plane containing color values of the same

type. An image unpacking process unpacks an input image into component arrays for input to the encoding process. The image unpacking process is not defined by the VC-5 essence standard. Likewise, a decoding process outputs an ordered set of component arrays that can be packed into an image by an image repacking process that is not defined by the VC-5 essence standard.

The sample encoder and reference decoder provided with the test materials include image unpacking and repacking code to enable testing with common image formats.

The test materials are described in section 6. The reference decoder for testing bitstreams created by implementations of a VC-5 encoder for compliance with the VC-5 essence standard is described in section 6.2.

The test materials include a sample encoder (section 6.3), a program for converting between image formats (section 6.4), a program for comparing image files (section 6.5), and the images used by the sample encoder to create the reference bitstreams used for conformance testing (section 6.6).

The image file formats used by the sample encoder, reference decoder, and utility programs are described in section 7. Procedures and scripts that can be used for conformance testing are described in section 8. The criteria for conformance to the VC-5 essence standard are specified in section 9. Instructions for installing the test materials are provided in section 10 and in a text file provided with the test materials distribution.

Instructions for installing and building the programs and contact information for submitting bug reports are provided in a separate document that is included in the test materials distribution.

A VC-5 bitstream can represent common image formats as well as Color Filter Array (CFA) images such as Bayer. The component values represented in a VC-5 bitstream can have up to 24 bits of precision. To accommodate the variety of images that can be encoded into a VC-5 bitstream, the software provided with the test materials can use images stored as unformatted binary files (section 7.2) or an ordered set of component arrays stored as one array per file with the channel number encoded into the pathname (section 7.3). For convenience, the software also supports the DPX file format (section 7.4).

Reference bitstreams for testing VC-5 decoder implementations for compliance with the VC-5 essence standard and the corresponding reference images decoded from the reference bitstreams by the reference decoder are listed in Annex A.

6 Description of the Test Materials

6.1 Source Code

The source code for the reference decoder and sample encoder is written in the C programming language according to ISO/IEC 9899:2011.

6.2 VC-5 Parts

This recommended practice covers parts of the VC-5 essence standard listed in the enabled part number column of Table 1.

Enabled Part Number	VC-5 Essence Standard Title	Implied Part Numbers
1	Elementary Bitstream	

Enabled Part Number	VC-5 Essence Standard Title	Implied Part Numbers
3	Image Formats	1
4	Subsampled Color Difference Components	1, 3
5	Layers	1, 3
6	Sections	1, 3

Table 1. Parts of the VC-5 essence standard.

The reference decoder and sample encoder allow different parts of the VC-5 essence standard to be tested. Parts are enabled through a combination of compile-time and runtime variables.

To use the capabilities defined in a part of the VC-5 essence standard, enable that part at both compile-time and runtime. A part can be enabled at compile-time using the `VC5_ENABLED_PARTS` compile-time variable in the file `$(ROOT)/common/include/config.h` that is provided with the source code distribution. A part can be enabled at runtime by using the parts command-line argument to explicitly enable that part. Some parts are implicitly enabled at runtime when another part is enabled (see implied part numbers in Table 1).

Part 1 is always enabled and does not need to be specified on the command-line. Specifying part 2 has no effect as there is no code associated with the conformance specification. Part 4 implies support for part 3 since ST 2073-4 requires features specified in ST 2073-3.

Parts 5 and 6 imply support for part 3 since it is expected that most VC-5 implementations will always use the image format features specified in part 3. This behavior can be changed by modifying compile-time variables in the file `$(ROOT)/common/include/config.h` (see the documentation provided with the test materials).

The reference decoder and sample encoder distributed with the test materials have parts 1, 3, 4, 5 and 6 enabled at compile-time by default. If the reference decoder and sample encoder are built with all parts enabled, then the same build can be used for testing any part by enabling that part at runtime using the parts command-line argument.

To encode a bitstream that includes syntax elements defined in VC-5 parts 3 and 4, run the encoder as follows:

```
encoder --parts 4 [other options] <image file> <bitstream file>
```

In this document, when the text says that a VC-5 part is enabled, it means that the part is enabled at both compile-time and runtime using the variables and command-line options described above.

6.3 Reference Decoder

6.3.1 General Usage

The reference decoder reads a bitstream that is compliant with the VC-5 essence standard and outputs one or more images. Each image is output as an unformatted file (section 7.2), an ordered set of component

array files (section 7.3), or a DPX file with 10-bit packed RGB color values (section 7.4). The reference decoder includes an implementation of an image repacking process that packs the component arrays output by the decoder into an output image.

To invoke the reference decoder from the command line:

```
decoder [options] <bitstream file> <image file>
```

Command-line options for the reference decoder are listed in Table 2.

All options present on the command line come before the input bitstream file on the command line.

The reference decoder provided with the test materials includes an implementation of an image repacking process to allow the decoder to output packed images. The packed image file formats that are supported by the reference decoder are listed in Table 7. The code for the image repacking process can be modified to support other image formats.

Option (Short or Long Format)	Description
-p <file format> --pixel <file format>	File format of the image input to the sample encoder to enable the decoder to output an image in the same format as input to the sample encoder (default is the file format of the output image)
-o <file format> --output <file format>	File format of the output image (default is the file format corresponding to the filename extension of the output image file)
-w <image width> --width <image width>	Width of the input image in samples (default is the width obtained from the bitstream)
-h <image height> --height <image height>	Height of the input image in sample rows (default is the height obtained from the bitstream)
-v --verbose	Enable verbose output during decoding (default is no verbose output)
-h --help	Print the program usage and command-line options (default is not to print the program usage text)
-P <parts list> --parts <parts list>	List of part numbers from Table 6, separated by commas without spaces, to enable the corresponding parts of the decoder (section 6.7)

Option (Short or Long Format)	Description
-S <sections list> --sections <sections list>	List of section numbers listed in Table 2, separated by commas, to enable processing of the specified sections during decoding

Table 2. Reference decoder command-line options.

The value of <file format> is case insensitive and can be any of the file formats listed in Table 7. The value of <image width> or <image height> can be any unsigned integer that corresponds to valid dimensions of the component arrays produced by the image unpacking process. For example, the minimum width or height of a component array is 48, which implies that the minimum width and height of an image in BYR4 or NV12 format is 96.

6.3.2 Layer Decoding

The image file command-line argument can be a string format specification generated a sequence of output filenames so that each decoded layer is stored in a separate file. For example, if the output image file is output-%04d.byr4 and the bitstream contains 4 layers, then the reference decoder outputs the layers in separate files as follows:

```
output-0001.byr4

output-0002.byr4

output-0003.byr4

output-0004.byr4
```

where %04d is replaced by the layer number.

If part 5 is enabled, then the reference decoder will output one image file per layer present in the bitstream using the image file provided on the command-line as a format specification string as described above.

If part 5 is not enabled, then the reference decoder will output the first layer present in the bitstream, storing the image in the filename specified by the image file provided as a command-line parameter, and ignore any remaining layers in the bitstream.

6.3.3 Section Decoding

If part 6 is enabled, then the reference decoder will output the type and size of each section specified by the sections command-line option that is encountered in the bitstream to the sections log file. The filename of the sections log file is the filename of the input bitstream file with the file extension replaced by "log".

If part 6 is not enabled, the reference decoder will ignore sections encountered in the bitstream.

Decoding is not affected by enabling or disabling sections, except for image sections as described in section 6.3.4.

6.3.4 Image Section Decoding

If part 6 is enabled and the sections command-line option specifies that image sections (section number 1) are to be processed, then the command-line arguments can include multiple output pathnames:

```
decoder [options] <bitstream file> <image file 1> <image file 2> … <image file n>
```

All command-line options precede the bitstream file command-line parameter. The bitstream contains the bitstream to be decoded. The remaining command-line arguments specify the output files for the decoded images. Output pathnames are used in the order in which image sections are present in the bitstream.

Typically, one image pathname is specified per image section in the bitstream, but the last output pathname can be a format specification string that generates pathnames for the output images. Only the last image pathname provided as a command-line argument can be a format string specification.

Providing a format specification string allows the decoder to output one image file for each image section encountered in the bitstream. If the last image pathname is not a format string specification, then the decoder will report an error if the number of image sections in the bitstream exceeds the number of image pathnames provided as command-line arguments.

6.4 Sample Encoder

6.4.1 General Usage

The sample encoder takes as input a single image that is either an unformatted file (section 7.2) or a DPX file with 10-bit packed RGB color values (section 7.4) and outputs a VC-5 compliant bitstream. The sample encoder includes an implementation of an image unpacking process for unpacking the input image into component arrays.

To invoke the sample encoder from the command line:

```
encoder [options] <image file> <bitstream file>
```

All options present on the command line come before the input image file on the command line.

Command-line options for the sample encoder are listed in Table 3. The image file can be one of the packed image file formats listed in Table 7. The encoded bitstream is placed in a binary file with the filename extension “vc5”.

Option (Short or Long Format)	Description
`-w <image width>` `--width <image width>`	Width of the image in samples (needed whenthe file format does not provide the width)
`-h <image height>` `--height <image height>`	Height of the image in sample rows (needed when the file format does not provide the height)
`-p <file format>` `--pixel <file format>`	File format of the image (default is the file format corresponding to the filename extension of the image file)
`-q Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8,Q9`	Quantization divisors for the highpass subbands (default values are compiled into the encoder)
`-v` `--verbose`	Enable verbose output during encoding (default is no verbose output)
`-h` `--help`	Print the program usage and command-line options (default is not to print the program usage text)
`-P <parts list>` `--parts <parts list>`	List of part numbers from Table 1, separated by commas without spaces, to enable the corresponding parts of the encoder (see section 6.7)
`-S <sections list>` `--sections <sections list>`	List of section numbers from Table 4, separated by commas without spaces, specifying the types of sections defined in ST 2073-6 that are encoded in the bitstream
`-L <image section layers>` `--layers <image section layers>`	Number of layers per image section separated by commas without spaces (see section 6.4.5)

Table 3. Sample encoder command-line options.

The unformatted image file formats (section 7.2) do not provide the image dimensions so the width and height are specified as command-line options. The file format can be inferred from the filename extension (Table 5) or passed as a command-line option. The sample encoder does not support using an ordered set of component arrays as an input format (section 7.3). A file format specified explicitly as a command-line option takes precedence over a file format inferred from the filename extension.

6.4.2 Layer Encoding

The default value for the layer count is 1.

If part 5 is enabled, then the sample encoder will encode input provided as command-line arguments, one filename per image.

If part 5 is not enabled, then the sample encoder will encode the one image file provided as a command-line argument into the bitstream without adding any of the bitstream elements specified by ST 2073-5.

6.4.3 Section Encoding

If part 6 is enabled, then

> If the sections command-line option is specified on the command line, then the sample encoder will insert the specified section elements into the bitstream for every section type defined in SMPTE ST 2073-6 and listed in Table 4.

If part 6 is not enabled, then

> If the sections command-line option is specified on the command line, then the sample encoder will report an error and not attempt to encode the bitstream.

Number	Section
1	Section element marking the beginning and extent of the portion of the bitstream containing the representation of an image
2	Section element marking the beginning and extent of the tag-value pairs that comprise the bitstream header
3	Section element marking the beginning and extent of the portion of the bitstream containing all syntax elements that are unique to a single layer (includes all channels in the layer)
4	Section element marking the beginning and extent of the portion of the bitstream containing all syntax elements that are unique to a single channel (includes all wavelets in the channel)
5	Section element marking the beginning and extent of the portion of the bitstream containing syntax elements that are unique to a single wavelet (includes all subbands in the wavelet)
6	Section element marking the beginning and extent of the portion of the bitstream containing tag-value pairs and the codeblock for a single subband

Table 4. Section type numbers and the corresponding section types.

6.4.4 Image Section Encoding

If part 6 is enabled and the sections command-line option specifies that image sections (section number 1) are to be processed, then the command-line arguments can include multiple input pathnames:

```
encoder [options] <image file 1> <image file 2> … <image file n> <bitstream file>
```

Each image in each file will be encoded as an image section in the order in which the image files are listed on the command line. The image file pathnames cannot be format specification strings. The last command-line argument is the output file for the encoded bitstream.

6.4.5 Encoding Image Sections with Layers

Image sections can contain layers within the image section. Each layer image within an image section has the same image dimensions, pixel format, and image encoding. The number of input images specified on the command line equals the total number of layers in all image sections. The partitioning of input images into layers and sections is specified by the layers command-line option using a comma-separated list (no spaces) of the number of layers in each image section. For example, encode two image sections, the first image section with 2 layers and a second image section with 3 layers, using the command

```
encoder -P 5,6 -S 1 -L 2,3 image1a image1b image2a image2b image2c bitstream.vc5
```

Both parts 5 and 6 (layers and sections) are enabled, image sections are enabled, and the number of images provided on the command-line equals the total number of layers specified by the layers option.

6.5 Image Converter

The image converter is a utility program that is provided as a convenience. The program can convert between several of the image formats used by the test materials.

To invoke the converter from the command line:

```
converter [options] <input image file> <output image file>
```

The converter command-line options are described in Table 5.

The file formats of the input and output images are inferred from the filename extensions, but can be overridden using command–line options.

Option (Short or Long Format)	Description
`-w <image width>` `--width <image width>`	Width of the input image in samples (needed when the file format does not provide the width)
`-h <image height>` `--height <image height>`	Height of the input image in sample rows (needed when the file format does not provide the height)
`-p <file format>` `--pixel <file format>`	File format of the input image (default is the file format inferred from the input filename extension)

Option (Short or Long Format)	Description
-o <file format> --output <file format>	File format of the output image (default is the file format inferred from the output filename extension)
-v --verbose	Enable verbose output during conversion (default is no verbose output)
-h --help	Print the program usage and command-line options (default is not to print the program usage text)

Table 5. Image converter command-line options.

6.6 Image Comparer

The image comparer is a C language utility program that is provided as a convenience. The program computes the PSNR between two images.

To invoke the comparer from the command line:

```
comparer [options] <image file 1> <image file 2>
```

The comparer command-line options are described in Table 6.

Option (Short or Long Format)	Description
-w <image width> --width <image width>	Width of both images in samples (required if the file format does not provide the width)
-h <image height> --height <image height>	Height of both images in sample rows (required if the file format does not provide the height)
-p <file format> --pixel <file format>	File format of both images (default is the filename extension)
-v --verbose	Enable verbose output during conversion (default is no verbose output)
-h --help	Print the program usage and command-line options (default is not to print the program usage text)

Table 6. Image comparer command-line options.

Both image files have the same image dimensions and pixel format.

6.7 Test Cases

The test materials include images organized into the directory structure specified in section 8.

Reference images for three test cases are provided in the test materials distribution: boxes, solid, and gradient. The boxes test case consists of synthetic images of randomly placed, overlapping rectangles with randomly chosen colors. The solid test case consists of synthetic images that are a single randomly chosen color. The gradient test case consists of synthetic images that are a single gray wedge from black to white.

The reference images that have width or height equal to 48 (or equal to 96 in the case of BYR4 and NV12) were chosen so that the component arrays output by the image unpacking process have the minimum width or height allowed by the VC-5 essence standard.

The reference images that have width or height equal to 49 (or equal to 97 in the case of BYR4 and NV12) were chosen so that the component arrays output by the image unpacking process have width or height that is an odd number. Wavelet transforms normally require an even input dimension, but ST 2073-1 specifies a mechanism for handling an odd width or height. The mechanism adds one to an odd dimension. In the case where the input width or height of the unpacked component arrays is 49, the corresponding dimension of each wavelet in the three level wavelet sequence will be 25, 13, and 7, respectively, so that the ability of a VC-5 encoder or decoder to handle odd wavelet dimensions will be exercised at each wavelet level.

6.8 Testing Layers

To enable testing of encoders that implement VC-5 Part 5 Layers, for each reference image listed in Section 6.6 an image is provided with the sequence number 0001, forming a pair of images that can be encoded as two layers in a single bitstream.

6.9 Testing Sections

Except for image sections (described in section 6.10), sections can be tested as part of testing any other part of the VC-5 standard by specifying part 6 on the command line (using the -p command-line option) along with the sections that are to be processed during decoding (using the -s command-line option)

If sections are enabled during decoding, then the decoder writes a log file containing information about the sections encountered while parsing the bitstream. The filename of the sections log file is the same as the filename of the input bitstream, with the file extension replaced by “log”. The sections log file is written to the same directory as the files containing the decoded images.

6.10 Testing Image Sections

A different folder structure is used for the reference bitstreams and image files for testing image sections. The sections directory is a sub-directory of the media sub-directory of the root directory into which the codec software was installed (see section 10). Each subdirectory of the sections directory corresponds to an image sections test and contains the images files encoded as image sections into a single bitstream as part of the test.

Encoded bitstream files and decoded image files are written into the results sub-directory of the sections sub-directory.

6.11 Testing Image Section Layers

A different folder structure is used for the reference bitstreams and image files for testing image sections that contain nested layers. The layers directory is a sub-directory of the sections directory (see section 6.10). The layers directory contains images that are input to the sample encoder to create a reference bitstream, the reference bitstream produced by the sample encoder, and the reference images produced by the reference decoder from the reference bitstream.

7 File Formats

7.1 Image File Formats

The layout of the images used for conformance testing can be described as a sample array (Figure 1). The file format specifies the order of the component values in each sample, the number of bits per component value, and any padding between component values.

The image data for the byr4 file format is a sample array using a 2 by 2 pattern element with the RGGB samples arranged in each pattern element as

R G
G B

Other images with 4:4:4(:4) sampling such as R'G'B'(A) or Y'C'$_B$C'$_R$(A) can be represented as sample arrays with a 1 by 1 pattern element.

Image data for the nv12 file format comprises a sample array for the 8-bit Y' values immediately followed by a sample array with interleaved 8-bit C'$_B$ and C'$_R$ values. The sample array of C'$_B$C'$_R$ values is the same width as the Y' sample array and half the height of the Y' sample array. See ISO/IEC 13818-2:2013.

By default, the file extension indicates the file format of the image file. The file extension can be lower case, upper case, or mixed case. The mapping from the lower case representation of the file extension to the file format is described in Table 7.

All image files used by the programs described in this document store the image in the file as an array of samples in raster-scan order.

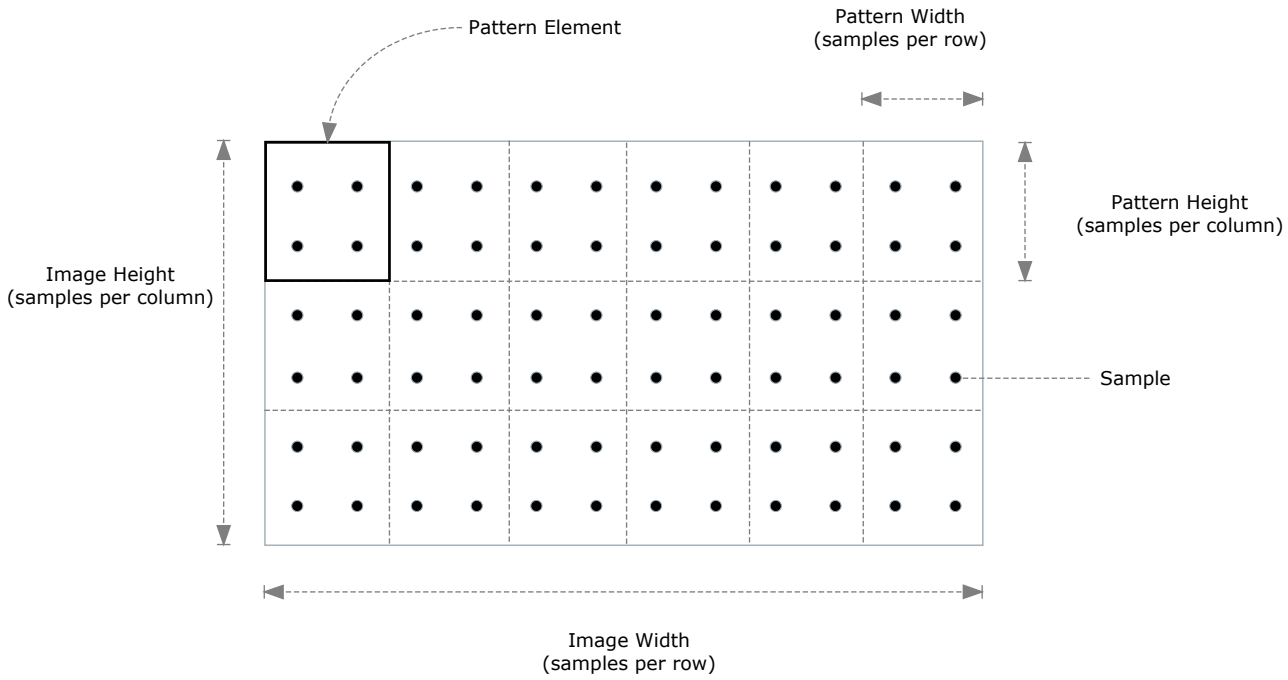

Figure 1. Sample array.

File Format	Description
byr4	Bayer pattern RGGB in a 2x2 pattern element with 16 bits per color value, each little-endian value in the range from 0 to 65535 inclusive (refer to section 7.1).
rg48	R, G, B color components in order with 16 bits per color value, each little-endian value in the range from 0 to 65535 inclusive.
b64a	A, R, G, B color components in order with 16 bits per color value, each big-endian value in the range from 0 to 65535 inclusive.
ca32	Ordered set of component array files, one file per channel. Each component array file consists of the component array values stored as little-endian 32-bit unsigned integers in raster-scan order with no file header. The pathname is a valid string format specification for inserting the channel number (section 7.3).
dpx	Image file compliant with SMPTE ST 268-1 and containing 10-bit RGB color values packed into a 32-bit word.
nv12	An array of Y', C'_B, and C'_R color components with 8 bits per color value and 4:2:0 color difference component subsampling (ISO/IEC 13818-2:2013).

Table 7. File formats used by the test materials.

With the exception of the DPX file format, all file formats listed in Table 7 are unformatted image files.

7.2 Unformatted Image Files

An unformatted image file contains the image samples stored in raster-scan order without any header. The number of samples per row is the width of the image without padding and the number of rows of samples is the height of the image without padding.

7.3 Component Array Files

The reference decoder can output an ordered set of component arrays with one component array per file. Each file contains the values of the component array as 32-bit unsigned integers in raster-scan order with no image header. The output pathname is a valid string format specification that allows the reference decoder to output multiple component array files, one per channel, with the channel number represented in the pathname. (See the documentation for the sprintf function in the standard C library). For example, if four component arrays are represented in the input bitstream and the output pathname is

```
boxes-%02d.ca32
```

then the reference decoder will output four files:

```
boxes-00.ca32

boxes-01.ca32

boxes-02.ca32

boxes-03.ca32
```

The dimensions of each component array are not represented in the unformatted component array files.

7.4 DPX Image Files

The DPX image format used in this specification is 10-bit RGB color values packed in a 32-bit word as indicated by the descriptor in the Image Information Header having the value 50. See SMPTE ST 268-1.

7.5 Other Image File Formats

The sample encoder includes a subroutine called ImageUnpackingProcess that can be modified to accept other input image formats and the reference decoder includes a subroutine called ImageRepackingProcess that can be modified to produce other output image formats from the decoded component arrays.

7.6 Bitstream File Formats

Files that contain VC-5 encoded bitstreams follow the naming conventions for image files with the exception that the part of VC-5 that was enabled is included in the pathname.

8 Conformance Testing (Informative)

8.1 Codec Test Script

The scripts directory contains scripts written in Tcl that invoke the sample encoder and reference decoder on images in the media directory. The Tcl test script is in the file `testcodec.tcl` in the directory `$(ROOT)/scripts`.

The media directory is organized into a hierarchy that follows naming conventions. The Tcl test script relies on these naming conventions. The media directory contains one subdirectory per test case. The directory for each test case contains one subdirectory for each unique image width and height. The image width and height directory contains one subdirectory for each file format. The file format directory contains one or more files with the image width, image height, and file format specified by the name of its parent directories.

If the media directory is moved to a different location or a different media directory is used for testing, the symbol media in the Tcl test script can be changed to the location of the media directory, or the new media directory can be specified as a command-line argument.

Suppose that the name of a test case is boxes, this test case contains images with dimensions of width 1920 by height 1080 and width 1280 by height 720, each image size includes images with file formats b64a and byr4, and there is one image file for each combination of image dimensions and file format, then there are four pathnames relative to the media directory for image files:

```
$(ROOT)/media/boxes/1920x1080/b64a/boxes-1920x1080-0000.b64a

$(ROOT)/media/boxes/1920x1080/byr4/boxes-1920x1080-0000.byr4

$(ROOT)/media/boxes/1280x720/b64a/boxes-1280x720-0000.b64a

$(ROOT)/media/boxes/1280x720/byr4/boxes-1280x720-0000.byr4
```

The directory names in the pathname specify the image dimensions and file format. Each image filename includes the name of the test case, the image dimensions, the sequence number that distinguishes multiple images within a single directory, and the filename extension that indicates the file format.

The Tcl test script traverses the directory tree for each test case. Some of the image file formats do not contain the image dimensions or file format, so the directory name and filename extension are used to determine the dimensions and format of the image. For example, the pathname

```
$(ROOT)/media/boxes/1920x1080/b64a/boxes-1920x1080-0001.b64a
```

specifies that the image width is 1920, the image height is 1080, and the file format is b64a as specified by either the filename extension or the name of the directory that contains the image file.

The Tcl test script obtains the image dimensions from the directory name, for example by splitting the directory name 1920x1080 and the letter “x” into the width and height. The width and height is passed to the sample encoder and reference decoder programs as command-line arguments (see Table 1 and Table 2). The sample encoder and reference decoder programs use the filename extension to determine the file format. The Tcl test script can be modified to use the file format in the pathname and pass this format to the sample encoder or reference decoder programs as command-line arguments.

The Tcl test script performs the following actions:

1. Traverses the hierarchical directory structure for each test case.
2. Creates a directory for test results in the test case directory (see below).
3. Invokes the sample encoder on images found in each test case and places the encoded bitstream in the results directory.
4. Invokes the reference decoder on each encoded bitstream and places the decoded image in the results directory.
5. If the test case contains a directory called master, then the Tcl test script compares the decoded image with the reference image in the master directory using a simple binary file comparison.

The directory for test results is a subdirectory path starting in the folder that contains the test media. The subdirectory path is formed from the location of the media directory, the name of the Tcl test script, the date, the part number supplied as a command-line argument to the test script, and the build configuration. For example:

```
$(ROOT)/media/boxes/results/testcodec-2015-05-04/part4/release
```

The Tcl test script reports the test cases that produced decoded images that do not exactly match the reference image. The comparison distinguishes between debug and release builds, because a debug build can insert extra information into the bitstream to assist in debugging.

The Tcl test script accepts the command-line arguments listed in Table 8. For example, to run the boxes and gradient test cases in the media directory installed along with the test materials using the debug build configuration of the sample encoder and reference decoder built using the make file in the root directory of the installed software:

```
testcodec.tcl –m ../media –b debug –t boxes,gradient
```

The sample encoder and reference decoder can be used without the test script by providing the parameters on the command line. For example, to encode one of the images provided with the test materials and decode the resulting bitstream:

```
encoder –w 1280 –h 720 boxes-1280x720-0000.byr4 boxes-1280x720-0000-byr4.vc5

decoder boxes-1280x720-0000-byr4.vc5 boxes-1280x720-0000-byr4-vc5.byr4
```

These example commands mimic the file naming conventions described in this section.

Option	Description
-r <root directory>	Top-level directory for the installation of the VC-5 test materials. (The default depends on the platform)

Option	Description
`-m <media directory>`	Top-level directory that contains the test cases. (The default is `$(ROOT)/media`)
`-b <configuration>`	Release or debug build configuration. (The default build configuration is release)
`-c <compiler>`	Tool used to build the encoder and decoder. The only documented value for the compiler is make which is the default value.
`-t <test cases>`	List of test cases to run, separated by commas without spaces
`-d`	Output a DPX file to provide a displayable picture of the decoded image. The pathname is derived from the pathname for the decoded image. The default is to not output a displayable picture.
`-v`	Verbose output (default is to suppress verbose output)
`-e`	Even more version output (for debugging). The default is to suppress extra output for debugging.
`-p <part>`	VC-5 part passed to the codec as a command-line parameter.
`-s <sections>`	Comma-separated list of section numbers enabled for part 6 encoding (only applies if part 6 is enabled).
`-l <logfile>`	Specify a log file for recording the work done by the test script.
`-a`	Perform all tests in the comprehensive test suite (see section 8.2)

Table 8. Tcl test script command-line options.

8.2 Comprehensive Test Suite

The Tcl script file `testcodec-suite.tcl` defines a set of tests called the comprehensive test suite. This set of tests is intended for performing a full set of regression tests on all codec capabilities. The regression tests can be run by executing the command:

```
testcodec.tcl -a
```

If the `-a` switch is provided on the command line, then the `-t`, `-p`, and `-s` switches are ignored.

9 Conformance Specification

9.1 Test Materials

The materials for testing conformance to the VC-5 essence standard shall comprise the following items:

1. The reference bitstream files listed in Annex A,
2. The release configuration of the VC-5 reference decoder built according to the instructions provided with the test materials distribution (section 10).
3. The reference decoded image files listed in Annex B,
4. The release configuration of the VC-5 sample encoder built according to the instructions provided with the test materials distribution (section 10).
5. Sample image files as described in section 6.6.

9.2 Bitstream Conformance

A conformance test for a VC-5 bitstream shall use the VC-5 reference decoder to verify conformance of the bitstream to the parts the VC-5 essence standard that are enabled in the reference decoder.

A bitstream shall be conformant to specified parts of the VC-5 essence standard if and only if all of the following conditions are satisfied:

1. The specified parts are enabled at compile-time when the reference decoder is built,
2. The specified parts are enabled at runtime when the reference decoder is run,
3. The bitstream conforms to the syntax and semantics of the specified parts of the VC-5 essence standard,
4. The reference decoder completes the decoding process for the bitstream without reporting any warnings or errors, and
5. The reference decoder successfully and completely produces an ordered set of component arrays with the proper dimensions.

The component arrays may be represented as one of the packed image formats listed in Table 5.

9.3 Decoder Conformance

A conformance test on an implementation of the decoding process for specified parts of the VC-5 essence standard shall:

1. Apply the decoder implementation to each of the reference bitstream files for the corresponding parts,
2. Enable the specified parts of the VC-5 essence standard at compile-time and runtime,
3. Decode each reference bitstream file to an ordered set of component arrays with the proper dimensions without reporting any warnings or errors,
4. Output a packed image file or ordered set of component arrays that equals the corresponding reference decoder image file or set of component arrays.

Two images files are equal if and only if the image files have the same length (in bytes) and the values of the bytes at the same offset in each file are equal. The ca32 file format is the only file format required for compliance testing.

10 Installing and Building the Test Materials

10.1 Test Materials Installation

The test materials are distributed by SMPTE. Install the test materials into an empty directory. This directory will be the root directory of the test materials distribution, designated in this document by the symbol `ROOT`.

10.2 Test Materials Contents

The test materials distribution installed in the root directory includes the following folders:

`$(ROOT)/common`

Source code that is common to both the sample encoder and reference decoder.

`$(ROOT)/encoder`

Source code for the sample encoder.

`$(ROOT)/decoder`

Source code for the reference decoder.

`$(ROOT)/tables`

Codebook used by the sample encoder and reference decoder.

`$(ROOT)/external`

Software developed by third parties that is used by the test materials.

`$(ROOT)/converter`

Source code for a C language program for converting between image file formats.

`$(ROOT)/comparer`

Source code for a C language program that can be used to compare the decoded image with the image that was input to the encoder. The comparer program is not currently used by the test scripts.

`$(ROOT)/scripts`

Scripts written in Tcl for testing the sample encoder and reference decoder.

`$(ROOT)/media`

Sample images and reference bitstreams that are compliant with the VC-5 essence standard.

10.3 Build Scripts and Documentation

The root directory of the test materials distribution contains the following files:

$(ROOT)/readme.html

$(ROOT)/readme.pdf

Detailed instructions on building and running the software in the test materials distribution in HTML and PDF formats.

$(ROOT)/release.html

$(ROOT)/release.pdf

Release notes in HTML and PDF formats.

$(ROOT)/Makefile

Top-level make file for building the debug and release configurations of the sample encoder and reference decoder. The top-level make file can also create detailed documentation from the source code. Refer to the readme file for details.

10.4 Test Materials Media

The media folder is divided into test cases with one folder per test case named after the test case:

$(ROOT)/media/boxes

Sample images and reference bitstreams for the “boxes” test case.

$(ROOT)/media/gradient

Sample images and reference bitstreams for the “gradient” test case.

$(ROOT)/media/solid

Sample images and reference bitstreams for the “solid” test case.

$(ROOT)/media/sections

Sample images and reference bitstreams for testing image sections as defined in VC-5 Part 6. The image sections test case differs from the other test cases due to the unique requirements for testing image sections (see section 6.10).

Each test case is divided into sub-directories for the image dimensions. For example, all image files for the boxes test case with dimensions 1280 by 720 are in the folder `$(ROOT)/media/boxes/1280x720`.

The sub-directory of each image dimension is divided into sub-directories by pixel format. For example, all image files for the boxes test case with dimensions 1920 by 1080 in the Bayer pixel format are in the folder `$(ROOT)/media/boxes/1280x720/byr4`.

Encoded bitstreams corresponding to the test images and the results of decoding the bitstreams are in the master folder for each test case, subdivided by VC-5 part and build configuration. For example, the folder `$(ROOT)/media/boxes/master/part4/release` contains the bitstreams and decoded image files for the boxes test case executed with the release configuration of the encoder and decoder for VC-5 part 4.

10.5 Building the Encoder and Decoder

The debug and release configurations of the sample encoder and reference decoder can be built by typing the command

```
make
```

in a terminal window open to the root directory of the test materials installation.

Other build options are described in the documents provided with the test materials distribution.

Annex A Reference Bitstreams (Normative)

A.1 Bitstream Files

The reference bitstreams included the test materials are listed in the following sections by VC-5 part number. The bitstreams are compliant with the part of the VC-5 essence standard indicated by the section heading and the pathname. The bitstreams were encoded using a release build of the sample encoder.

The reference bitstream files are located in the master sub-directory in each test case organized by VC-5 part number and build configuration. For example, the master bitstream files for the boxes test case, VC-5 part 1, release configuration, are in the directory `$(ROOT)/media/boxes/master/part1/release`.

A.2 VC-5 Part 1 Bitstreams

`boxes-1280x720-0000-part1-1280x720-byr4.vc5`

`boxes-1280x720-0000-part1-1280x720-rg48.vc5`

`boxes-1280x720-0001-part1-1280x720-byr4.vc5`

`boxes-1280x720-0001-part1-1280x720-rg48.vc5`

`boxes-1920x1080-0000-part1-1920x1080-byr4.vc5`

`boxes-1920x1080-0000-part1-1920x1080-rg48.vc5`

`boxes-1920x1080-0001-part1-1920x1080-byr4.vc5`

`boxes-1920x1080-0001-part1-1920x1080-rg48.vc5`

`boxes-640x480-0000-part1-640x480-byr4.vc5`

`boxes-640x480-0000-part1-640x480-rg48.vc5`

`boxes-640x480-0001-part1-640x480-byr4.vc5`

`boxes-640x480-0001-part1-640x480-rg48.vc5`

`gradient-1280x720-0000-part1-1280x720-byr4.vc5`

`gradient-1280x720-0001-part1-1280x720-byr4.vc5`

`gradient-640x480-0000-part1-640x480-b64a.vc5`

`gradient-640x480-0001-part1-640x480-b64a.vc5`

solid-1280x720-0000-part1-1280x720-byr4.vc5

solid-1280x720-0001-part1-1280x720-byr4.vc5

solid-48x48-0000-part1-48x48-b64a.vc5

solid-48x48-0000-part1-48x48-rg48.vc5

solid-48x48-0001-part1-48x48-b64a.vc5

solid-48x48-0001-part1-48x48-rg48.vc5

solid-48x49-0000-part1-48x49-rg48.vc5

solid-48x49-0001-part1-48x49-rg48.vc5

solid-49x48-0000-part1-49x48-rg48.vc5

solid-49x48-0001-part1-49x48-rg48.vc5

solid-49x49-0000-part1-49x49-rg48.vc5

solid-49x49-0001-part1-49x49-rg48.vc5

solid-640x480-0000-part1-640x480-rg48.vc5

solid-640x480-0001-part1-640x480-rg48.vc5

solid-96x96-0000-part1-96x96-byr4.vc5

solid-96x96-0001-part1-96x96-byr4.vc5

solid-96x97-0000-part1-96x97-byr4.vc5

solid-96x97-0001-part1-96x97-byr4.vc5

solid-97x96-0000-part1-97x96-byr4.vc5

solid-97x96-0001-part1-97x96-byr4.vc5

solid-97x97-0000-part1-97x97-byr4.vc5

solid-97x97-0001-part1-97x97-byr4.vc5

A.3 VC-5 Part 3 Bitstreams

boxes-1280x720-0000-part3-1280x720-byr4.vc5

boxes-1280x720-0000-part3-1280x720-rg48.vc5

boxes-1280x720-0001-part3-1280x720-byr4.vc5

boxes-1280x720-0001-part3-1280x720-rg48.vc5

boxes-1920x1080-0000-part3-1920x1080-byr4.vc5

boxes-1920x1080-0000-part3-1920x1080-rg48.vc5

boxes-1920x1080-0001-part3-1920x1080-byr4.vc5

boxes-1920x1080-0001-part3-1920x1080-rg48.vc5

boxes-640x480-0000-part3-640x480-byr4.vc5

boxes-640x480-0000-part3-640x480-rg48.vc5

boxes-640x480-0001-part3-640x480-byr4.vc5

boxes-640x480-0001-part3-640x480-rg48.vc5

gradient-1280x720-0000-part3-1280x720-byr4.vc5

gradient-1280x720-0001-part3-1280x720-byr4.vc5

gradient-640x480-0000-part3-640x480-b64a.vc5

gradient-640x480-0001-part3-640x480-b64a.vc5

solid-1280x720-0000-part3-1280x720-byr4.vc5

solid-1280x720-0001-part3-1280x720-byr4.vc5

solid-48x48-0000-part3-48x48-b64a.vc5

solid-48x48-0000-part3-48x48-rg48.vc5

solid-48x48-0001-part3-48x48-b64a.vc5

solid-48x48-0001-part3-48x48-rg48.vc5

solid-48x49-0000-part3-48x49-rg48.vc5

solid-48x49-0001-part3-48x49-rg48.vc5

solid-49x48-0000-part3-49x48-rg48.vc5

solid-49x48-0001-part3-49x48-rg48.vc5

solid-49x49-0000-part3-49x49-rg48.vc5

solid-49x49-0001-part3-49x49-rg48.vc5

solid-640x480-0000-part3-640x480-rg48.vc5

solid-640x480-0001-part3-640x480-rg48.vc5

solid-96x96-0000-part3-96x96-byr4.vc5

solid-96x96-0001-part3-96x96-byr4.vc5

solid-96x97-0000-part3-96x97-byr4.vc5

solid-96x97-0001-part3-96x97-byr4.vc5

solid-97x96-0000-part3-97x96-byr4.vc5

solid-97x96-0001-part3-97x96-byr4.vc5

solid-97x97-0000-part3-97x97-byr4.vc5

solid-97x97-0001-part3-97x97-byr4.vc5

A.4 VC-5 Part 4 Bitstreams

boxes-1280x720-0000-part4-1280x720-nv12.vc5

boxes-1280x720-0001-part4-1280x720-nv12.vc5

boxes-1920x1080-0000-part4-1920x1080-nv12.vc5

boxes-1920x1080-0001-part4-1920x1080-nv12.vc5

boxes-640x480-0000-part4-640x480-nv12.vc5

boxes-640x480-0001-part4-640x480-nv12.vc5

boxes-96x96-0000-part4-96x96-nv12.vc5

boxes-96x96-0001-part4-96x96-nv12.vc5

gradient-1280x720-0000-part4-1280x720-nv12.vc5

gradient-1280x720-0001-part4-1280x720-nv12.vc5

gradient-96x96-0000-part4-96x96-nv12.vc5

gradient-96x96-0001-part4-96x96-nv12.vc5

solid-1280x720-0000-part4-1280x720-nv12.vc5

solid-1280x720-0001-part4-1280x720-nv12.vc5

solid-96x96-0000-part4-96x96-nv12.vc5

solid-96x96-0001-part4-96x96-nv12.vc5

A.5 VC-5 Part 5 Bitstreams

The VC-5 Part 5 bitstreams were encoded from all images with the same image dimensions and pixel format within each test case with VC-5 part 5 enabled.

boxes-part5-1280x720-byr4.vc5

boxes-part5-1280x720-rg48.vc5

boxes-part5-1920x1080-byr4.vc5

boxes-part5-1920x1080-rg48.vc5

boxes-part5-640x480-byr4.vc5

boxes-part5-640x480-rg48.vc5

gradient-part5-1280x720-byr4.vc5

gradient-part5-640x480-b64a.vc5

solid-part5-1280x720-byr4.vc5

solid-part5-48x48-b64a.vc5

solid-part5-48x48-rg48.vc5

solid-part5-48x49-rg48.vc5

solid-part5-49x48-rg48.vc5

solid-part5-49x49-rg48.vc5

solid-part5-640x480-rg48.vc5

solid-part5-96x96-byr4.vc5

solid-part5-96x97-byr4.vc5

solid-part5-97x96-byr4.vc5

solid-part5-97x97-byr4.vc5

A.6 VC-5 Part 6 Bitstreams

A.6.1 VC-5 Part 6 Test Cases

The VC-5 Part 6 bitstreams were encoded from the same images used to create the VC-5 Part 1 bitstreams, but with VC-5 Part 6 enabled and section numbers 2 through 9 enabled.

Two sets of reference bitstreams are provided:

1. Bitstreams created with all sections except the image section, enabled at runtime by specifying section types 2 through 6, inclusive, using the sections command-line argument (see Table 3).
2. Bitstreams created from two images with different dimensions using the image section type specified using the sections command-line argument (see Table 3).

A.6.2 Non-Image Section Bitstreams

Reference bitstreams encoded from the corresponding reference images as determined by the file naming conventions used in this recommended practice with all section types enabled except the image section type:

boxes-1280x720-0000-part6-1280x720-byr4.vc5

boxes-1280x720-0000-part6-1280x720-rg48.vc5

boxes-1280x720-0001-part6-1280x720-byr4.vc5

boxes-1280x720-0001-part6-1280x720-rg48.vc5

boxes-1920x1080-0000-part6-1920x1080-byr4.vc5

boxes-1920x1080-0000-part6-1920x1080-rg48.vc5

boxes-1920x1080-0001-part6-1920x1080-byr4.vc5

boxes-1920x1080-0001-part6-1920x1080-rg48.vc5

boxes-640x480-0000-part6-640x480-byr4.vc5

boxes-640x480-0000-part6-640x480-rg48.vc5

boxes-640x480-0001-part6-640x480-byr4.vc5

boxes-640x480-0001-part6-640x480-rg48.vc5

gradient-1280x720-0000-part6-1280x720-byr4.vc5

gradient-1280x720-0001-part6-1280x720-byr4.vc5

gradient-640x480-0000-part6-640x480-b64a.vc5

gradient-640x480-0001-part6-640x480-b64a.vc5

solid-1280x720-0000-part6-1280x720-byr4.vc5

solid-1280x720-0001-part6-1280x720-byr4.vc5

solid-48x48-0000-part6-48x48-b64a.vc5

solid-48x48-0000-part6-48x48-rg48.vc5

solid-48x48-0001-part6-48x48-b64a.vc5

solid-48x48-0001-part6-48x48-rg48.vc5

solid-48x49-0000-part6-48x49-rg48.vc5

solid-48x49-0001-part6-48x49-rg48.vc5

solid-49x48-0000-part6-49x48-rg48.vc5

solid-49x48-0001-part6-49x48-rg48.vc5

solid-49x49-0000-part6-49x49-rg48.vc5

solid-49x49-0001-part6-49x49-rg48.vc5

solid-640x480-0000-part6-640x480-rg48.vc5

solid-640x480-0001-part6-640x480-rg48.vc5

solid-96x96-0000-part6-96x96-byr4.vc5

solid-96x96-0001-part6-96x96-byr4.vc5

solid-96x97-0000-part6-96x97-byr4.vc5

solid-96x97-0001-part6-96x97-byr4.vc5

solid-97x96-0000-part6-97x96-byr4.vc5

```
solid-97x96-0001-part6-97x96-byr4.vc5

solid-97x97-0000-part6-97x97-byr4.vc5

solid-97x97-0001-part6-97x97-byr4.vc5
```

A.6.3 Image Section Bitstreams

Image sections require a different directory structure for organizing the test media since each image section can represent an image with different dimensions and pixel format.

The folder `$(ROOT)/media/sections` is subdivided into separate directories for each image sections test case: boxes and mixed. Each image sections test case contains the set of images that will be encoded into a single bitstream with each image in a separate image section in the bitstream creating one bitstream for each image sections test case. The bitstreams are located in the master folder within the image sections test case:

The two bitstreams for image sections testing included in the test materials are:

```
$(ROOT)/media/sections/master/boxes/release/boxes.vc5
$(ROOT)/media/sections/master/mixed/release/mixed.vc5
```

A.6.4 Image Section Layer Bitstreams

Image sections that contain layers require a different directory structure for organizing the test media.

The bitstream created by the sample encoder to test image sections with nested layers is as follows:

```
$(ROOT)/media/sections/layers/boxes-section-layers-rg48.vc5
```

Annex B Reference Images (Normative)

B.1 Image Files

The test materials include the image files that were decoded using a release build of the reference decoder from the reference bitstreams.

The decoded image files are located in the master sub-directory in each test case organized by VC-5 part number and build configuration. For example, the decoded image files for the boxes test case, VC-5 part 1, release configuration, are in the directory `$(ROOT)/media/boxes/master/part1/release`.

Decoded image files with subsampled color difference components (nv12) are only relevant for VC-5 Part 4.

B.2 VC-5 Part 1 Images

```
boxes-1280x720-0000-part1-1280x720-byr4-vc5.byr4

boxes-1280x720-0000-part1-1280x720-rg48-vc5.rg48

boxes-1280x720-0001-part1-1280x720-byr4-vc5.byr4

boxes-1280x720-0001-part1-1280x720-rg48-vc5.rg48

boxes-1920x1080-0000-part1-1920x1080-byr4-vc5.byr4

boxes-1920x1080-0000-part1-1920x1080-rg48-vc5.rg48

boxes-1920x1080-0001-part1-1920x1080-byr4-vc5.byr4

boxes-1920x1080-0001-part1-1920x1080-rg48-vc5.rg48

boxes-640x480-0000-part1-640x480-byr4-vc5.byr4

boxes-640x480-0000-part1-640x480-rg48-vc5.rg48

boxes-640x480-0001-part1-640x480-byr4-vc5.byr4

boxes-640x480-0001-part1-640x480-rg48-vc5.rg48

gradient-1280x720-0000-part1-1280x720-byr4-vc5.byr4

gradient-1280x720-0001-part1-1280x720-byr4-vc5.byr4

gradient-640x480-0000-part1-640x480-b64a-vc5.b64a

gradient-640x480-0001-part1-640x480-b64a-vc5.b64a
```

solid-1280x720-0000-part1-1280x720-byr4-vc5.byr4

solid-1280x720-0001-part1-1280x720-byr4-vc5.byr4

solid-48x48-0000-part1-48x48-b64a-vc5.b64a

solid-48x48-0000-part1-48x48-rg48-vc5.rg48

solid-48x48-0001-part1-48x48-b64a-vc5.b64a

solid-48x48-0001-part1-48x48-rg48-vc5.rg48

solid-48x49-0000-part1-48x49-rg48-vc5.rg48

solid-48x49-0001-part1-48x49-rg48-vc5.rg48

solid-49x48-0000-part1-49x48-rg48-vc5.rg48

solid-49x48-0001-part1-49x48-rg48-vc5.rg48

solid-49x49-0000-part1-49x49-rg48-vc5.rg48

solid-49x49-0001-part1-49x49-rg48-vc5.rg48

solid-640x480-0000-part1-640x480-rg48-vc5.rg48

solid-640x480-0001-part1-640x480-rg48-vc5.rg48

solid-96x96-0000-part1-96x96-byr4-vc5.byr4

solid-96x96-0001-part1-96x96-byr4-vc5.byr4

solid-96x97-0000-part1-96x97-byr4-vc5.byr4

solid-96x97-0001-part1-96x97-byr4-vc5.byr4

solid-97x96-0000-part1-97x96-byr4-vc5.byr4

solid-97x96-0001-part1-97x96-byr4-vc5.byr4

solid-97x97-0000-part1-97x97-byr4-vc5.byr4

solid-97x97-0001-part1-97x97-byr4-vc5.byr4

B.3 VC-5 Part 3 Images

boxes-1280x720-0000-part3-1280x720-byr4-vc5.byr4

boxes-1280x720-0000-part3-1280x720-rg48-vc5.rg48

boxes-1280x720-0001-part3-1280x720-byr4-vc5.byr4

boxes-1280x720-0001-part3-1280x720-rg48-vc5.rg48

boxes-1920x1080-0000-part3-1920x1080-byr4-vc5.byr4

boxes-1920x1080-0000-part3-1920x1080-rg48-vc5.rg48

boxes-1920x1080-0001-part3-1920x1080-byr4-vc5.byr4

boxes-1920x1080-0001-part3-1920x1080-rg48-vc5.rg48

boxes-640x480-0000-part3-640x480-byr4-vc5.byr4

boxes-640x480-0000-part3-640x480-rg48-vc5.rg48

boxes-640x480-0001-part3-640x480-byr4-vc5.byr4

boxes-640x480-0001-part3-640x480-rg48-vc5.rg48

gradient-1280x720-0000-part3-1280x720-byr4-vc5.byr4

gradient-1280x720-0001-part3-1280x720-byr4-vc5.byr4

gradient-640x480-0000-part3-640x480-b64a-vc5.b64a

gradient-640x480-0001-part3-640x480-b64a-vc5.b64a

solid-1280x720-0000-part3-1280x720-byr4-vc5.byr4

solid-1280x720-0001-part3-1280x720-byr4-vc5.byr4

solid-48x48-0000-part3-48x48-b64a-vc5.b64a

solid-48x48-0000-part3-48x48-rg48-vc5.rg48

solid-48x48-0001-part3-48x48-b64a-vc5.b64a

solid-48x48-0001-part3-48x48-rg48-vc5.rg48

solid-48x49-0000-part3-48x49-rg48-vc5.rg48

solid-48x49-0001-part3-48x49-rg48-vc5.rg48

solid-49x48-0000-part3-49x48-rg48-vc5.rg48

solid-49x48-0001-part3-49x48-rg48-vc5.rg48

solid-49x49-0000-part3-49x49-rg48-vc5.rg48

solid-49x49-0001-part3-49x49-rg48-vc5.rg48

solid-640x480-0000-part3-640x480-rg48-vc5.rg48

solid-640x480-0001-part3-640x480-rg48-vc5.rg48

solid-96x96-0000-part3-96x96-byr4-vc5.byr4

solid-96x96-0001-part3-96x96-byr4-vc5.byr4

solid-96x97-0000-part3-96x97-byr4-vc5.byr4

solid-96x97-0001-part3-96x97-byr4-vc5.byr4

solid-97x96-0000-part3-97x96-byr4-vc5.byr4

solid-97x96-0001-part3-97x96-byr4-vc5.byr4

solid-97x97-0000-part3-97x97-byr4-vc5.byr4

solid-97x97-0001-part3-97x97-byr4-vc5.byr4

B.4 VC-5 Part 4 Images

boxes-1280x720-0000-part4-1280x720-nv12-vc5.nv12

boxes-1280x720-0001-part4-1280x720-nv12-vc5.nv12

boxes-1920x1080-0000-part4-1920x1080-nv12-vc5.nv12

boxes-1920x1080-0001-part4-1920x1080-nv12-vc5.nv12

boxes-640x480-0000-part4-640x480-nv12-vc5.nv12

boxes-640x480-0001-part4-640x480-nv12-vc5.nv12

boxes-96x96-0000-part4-96x96-nv12-vc5.nv12

boxes-96x96-0001-part4-96x96-nv12-vc5.nv12

gradient-1280x720-0000-part4-1280x720-nv12-vc5.nv12

gradient-1280x720-0001-part4-1280x720-nv12-vc5.nv12

gradient-96x96-0000-part4-96x96-nv12-vc5.nv12

gradient-96x96-0001-part4-96x96-nv12-vc5.nv12

solid-1280x720-0000-part4-1280x720-nv12-vc5.nv12

solid-1280x720-0001-part4-1280x720-nv12-vc5.nv12

solid-96x96-0000-part4-96x96-nv12-vc5.nv12

solid-96x96-0001-part4-96x96-nv12-vc5.nv12

B.5 VC-5 Part 5 Images

A decoder compliant with VC-5 Part 5 will output both layers as separate images. A decoder that is not complaint with VC-5 Part 5 will output the image corresponding to the first layer encountered in the bitstream.

boxes-1280x720-0000-part5-1280x720-byr4-vc5.byr4

boxes-1280x720-0000-part5-1280x720-rg48-vc5.rg48

boxes-1280x720-0001-part5-1280x720-byr4-vc5.byr4

boxes-1280x720-0001-part5-1280x720-rg48-vc5.rg48

boxes-1920x1080-0000-part5-1920x1080-byr4-vc5.byr4

boxes-1920x1080-0000-part5-1920x1080-rg48-vc5.rg48

boxes-1920x1080-0001-part5-1920x1080-byr4-vc5.byr4

boxes-1920x1080-0001-part5-1920x1080-rg48-vc5.rg48

boxes-640x480-0000-part5-640x480-byr4-vc5.byr4

boxes-640x480-0000-part5-640x480-rg48-vc5.rg48

boxes-640x480-0001-part5-640x480-byr4-vc5.byr4

boxes-640x480-0001-part5-640x480-rg48-vc5.rg48

gradient-1280x720-0000-part5-1280x720-byr4-vc5.byr4

gradient-1280x720-0001-part5-1280x720-byr4-vc5.byr4

gradient-640x480-0000-part5-640x480-b64a-vc5.b64a

gradient-640x480-0001-part5-640x480-b64a-vc5.b64a

solid-1280x720-0000-part5-1280x720-byr4-vc5.byr4

solid-1280x720-0001-part5-1280x720-byr4-vc5.byr4

solid-48x48-0000-part5-48x48-b64a-vc5.b64a

solid-48x48-0000-part5-48x48-rg48-vc5.rg48

solid-48x48-0001-part5-48x48-b64a-vc5.b64a

solid-48x48-0001-part5-48x48-rg48-vc5.rg48

solid-48x49-0000-part5-48x49-rg48-vc5.rg48

solid-48x49-0001-part5-48x49-rg48-vc5.rg48

solid-49x48-0000-part5-49x48-rg48-vc5.rg48

solid-49x48-0001-part5-49x48-rg48-vc5.rg48

solid-49x49-0000-part5-49x49-rg48-vc5.rg48

solid-49x49-0001-part5-49x49-rg48-vc5.rg48

solid-640x480-0000-part5-640x480-rg48-vc5.rg48

solid-640x480-0001-part5-640x480-rg48-vc5.rg48

solid-96x96-0000-part5-96x96-byr4-vc5.byr4

solid-96x96-0001-part5-96x96-byr4-vc5.byr4

solid-96x97-0000-part5-96x97-byr4-vc5.byr4

solid-96x97-0001-part5-96x97-byr4-vc5.byr4

solid-97x96-0000-part5-97x96-byr4-vc5.byr4

solid-97x96-0001-part5-97x96-byr4-vc5.byr4

solid-97x97-0000-part5-97x97-byr4-vc5.byr4

solid-97x97-0001-part5-97x97-byr4-vc5.byr4

B.6 VC-5 Part 6 Images

B.6.1 Non-Image Section Images

The output images from a VC-5 Part 6 compliant decoder are not provided since the resulting images will be identical to the output of a decoder that is not compliant with VC-5 Part 6.

The VC-5 reference decoder creates a log file with the type and size of each section encountered in the bitstream, in bitstream order. The log files corresponding to the bitstreams listed in Annex A.6 are as listed below:

boxes-1280x720-0000-part6-1280x720-byr4.log

boxes-1280x720-0000-part6-1280x720-rg48.log

boxes-1280x720-0001-part6-1280x720-byr4.log

boxes-1280x720-0001-part6-1280x720-rg48.log

boxes-1920x1080-0000-part6-1920x1080-byr4.log

boxes-1920x1080-0000-part6-1920x1080-rg48.log

boxes-1920x1080-0001-part6-1920x1080-byr4.log

boxes-1920x1080-0001-part6-1920x1080-rg48.log

boxes-640x480-0000-part6-640x480-byr4.log

boxes-640x480-0000-part6-640x480-rg48.log

boxes-640x480-0001-part6-640x480-byr4.log

boxes-640x480-0001-part6-640x480-rg48.log

gradient-1280x720-0000-part6-1280x720-byr4.log

```
gradient-1280x720-0001-part6-1280x720-byr4.log
gradient-640x480-0000-part6-640x480-b64a.log
gradient-640x480-0001-part6-640x480-b64a.log
solid-1280x720-0000-part6-1280x720-byr4.log
solid-1280x720-0001-part6-1280x720-byr4.log
solid-48x48-0000-part6-48x48-b64a.log
solid-48x48-0000-part6-48x48-rg48.log
solid-48x48-0001-part6-48x48-b64a.log
solid-48x48-0001-part6-48x48-rg48.log
solid-48x49-0000-part6-48x49-rg48.log
solid-48x49-0001-part6-48x49-rg48.log
solid-49x48-0000-part6-49x48-rg48.log
solid-49x48-0001-part6-49x48-rg48.log
solid-49x49-0000-part6-49x49-rg48.log
solid-49x49-0001-part6-49x49-rg48.log
solid-640x480-0000-part6-640x480-rg48.log
solid-640x480-0001-part6-640x480-rg48.log
solid-96x96-0000-part6-96x96-byr4.log
solid-96x96-0001-part6-96x96-byr4.log
solid-96x97-0000-part6-96x97-byr4.log
solid-96x97-0001-part6-96x97-byr4.log
solid-97x96-0000-part6-97x96-byr4.log
solid-97x96-0001-part6-97x96-byr4.log
```

```
solid-97x97-0000-part6-97x97-byr4.log

solid-97x97-0001-part6-97x97-byr4.log
```

B.6.2 Image Section Images

The decoded images for each image sections test case are as follows:

```
$(ROOT)/media/sections/boxes/boxes-1280x720-0000.rg48

$(ROOT)/media/sections/boxes/boxes-640x480-0000.rg48

$(ROOT)/media/sections/boxes/boxes-640x480-0001.rg48

$(ROOT)/media/sections/mixed/boxes-1920x1080-0000.byr4

$(ROOT)/media/sections/mixed/boxes-1920x1080-0000.rg48

$(ROOT)/media/sections/mixed/boxes-1920x1080-0001.byr4

$(ROOT)/media/sections/mixed/boxes-1920x1080-0001.rg48
```

B.6.3 Image Section Layer Images

The decoded images for the image section layers test case are as follows:

```
boxes-1280x720-0000-rg48-vc5.rg48

boxes-640x480-0000-rg48-vc5.rg48

boxes-640x480-0001-rg48-vc5.rg48
```

Bibliography

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

ISO/IEC 13818-2:2013 Information technology -- Generic coding of moving pictures and associated audio information -- Part 2: Video

ISO/IEC 9899:2011 Programming Language C (C11)

John K. Ousterhout and Ken Jones, Tcl and the Tk Toolkit (2nd Edition), Addison-Wesley, 2009