SMPTE RP 2073-2:2015
Revision of SMPTE RP 2073-2:2014

SMPTE RECOMMENDED PRACTICE

VC-5 Video Essence — Part 2: Conformance Specification

Table of Contents

Approved
October 9, 2015

Figures

Tables

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.

SMPTE RP 2073-2 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, and ST 2073-4.

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:

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

2. 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.

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.

This document does not include 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 2073-1:2013, 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 2.0 or later

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

4 Terms and Definitions

4.1 little-endian

Order of bytes that comprise a number with less significant bytes before more significant bytes.

4.2 VC-5 essence standard

SMPTE standards designated SMPTE ST 2073-1, ST 2073-3, and ST 2073-4.

5 Overview (Informative)

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

1. Source code for a reference decoder;
2. Source code for a sample encoder;
3. Utility programs and scripts useful for testing VC-5 decoder and encoder implementations;
4. Reference bitstreams for testing conformance of a decoder implementation to the VC-5 essence standard;
5. 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 Materials

6.1 Source Code

The source code for the reference decoder and sample encoder is written in the C programming language.

6.2 Reference Decoder

The reference decoder reads a bitstream that is compliant with the VC-5 essence standard and outputs a single image that is either 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 1.

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 6. The code for the image repacking process can be modified to support other image formats.

Table 1 – Reference decoder command-line options

Option (Short or Long Format)	Description
-p FileFormat --pixel FileFormat	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 FileFormat --output FileFormat	File format of the output image (default is the file format corresponding to the filename extension of the output image file)
-w ImageWidth --width ImageWidth	Width of the input image in samples (default is the width obtained from the bitstream)
-h ImageHeight --height ImageHeight	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 of part numbers from Table 5, separated by commas without spaces, to enable the corresponding parts of the decoder (Section 6.7)

The value of FileFormat is case insensitive and can be any of the file formats listed in Table 6. The value of ImageWidth or ImageHeight 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 Sample Encoder

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

Command-line options for the sample encoder are listed in Table 2. The image file can be one of the packed image file formats listed in Table 6. The encoded bitstream is placed in a binary file with the filename extension “vc5”.

Table 2 – Sample encoder command-line options

Option (Short or Long Format)	Description
-w ImageWidth --width ImageWidth	Width of the image in samples (required if the file format does not provide the width)
-h ImageHeight --height ImageHeight	Height of the image in sample rows (required if the file format does not provide the height)
-p FileFormat --pixel FileFormat	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 of part numbers from Table 5, separated by commas without spaces, to enable the corresponding parts of the encoder (Section 6.7)

The unformatted image file formats (Section 7.2) do not provide the image dimensions so the width and height must be provided 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 provided explicitly as a command-line option takes precedence over a file format inferred from the filename extension.

6.4 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 3.

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

Table 3 – Image converter command-line options

Option (Short or Long Format)	Description
-w ImageWidth --width ImageWidth	Width of the input image in samples (required if the file format does not provide the width)
-h ImageHeight --height ImageHeight	Height of the input image in sample rows (required if the file format does not provide the height)
-p FileFormat --pixel FileFormat	File format of the input image (default is the file format inferred from the input filename extension)
-o FileFormat --output FileFormat	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)

6.5 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 4.

Table 4 – Image comparer command-line options

Option (Short or Long Format)	Description
-w ImageWidth --width ImageWidth	Width of both images in samples (required if the file format does not provide the width)
-h ImageHeight --height ImageHeight	Height of both images in sample rows (required if the file format does not provide the height)
-p FileFormat --pixel FileFormat	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)

Both image files must have the same format due to limitations in the comparer program.

6.6 Test Images

The test materials include the following sample images organized into the directory structure specified in Section 8. The name of the test case is the prefix of the filename preceding the image dimensions.

```
boxes-1280x720-0000.byr4
boxes-1920x1080-0000.byr4
boxes-640x480-0000.byr4
boxes-1280x720-0000.rg48
boxes-1920x1080-0000.rg48
boxes-640x480-0000.rg48
boxes-1280x720-0000.nv12
boxes-1920x1080-0000.nv12
boxes-640x480-0000.nv12
boxes-96x96-0000.nv12
gradient-640x480-0000.b64a
gradient-1280x720-0000.byr4
gradient-1280x720-0000.nv12
gradient-96x96-0000.nv12
solid-48x48-0000.b64a
solid-1280x720-0000.byr4
solid-96x96-0000.byr4
solid-96x97-0000.byr4
solid-97x96-0000.byr4
solid-97x97-0000.byr4
solid-48x48-0000.rg48
solid-48x49-0000.rg48
solid-49x48-0000.rg48
solid-49x49-0000.rg48
solid-640x480-0000.rg48
solid-1280x720-0000.nv12
solid-96x96-0000.nv12
```

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 test 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 test 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 SMPTE 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.

6.7 Parts of the VC-5 Essence Standard

This recommended practice covers the parts of the VC-5 essence standard listed in Table 5.

Table 5 – Parts of the VC-5 essence standard

Part Number	Title	Implied Part Numbers
1	Elementary Bitstream	
3	Image Formats	1
4	Subsampled Color Difference Components	1, 3

The reference decoder and sample encoder allow different parts of the VC-5 essence standard to be tested. Parts are enabled through the 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 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 5).

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.

The reference decoder and sample encoder distributed with the test materials have parts 1, 3, and 4 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
```

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.

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 6.

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.

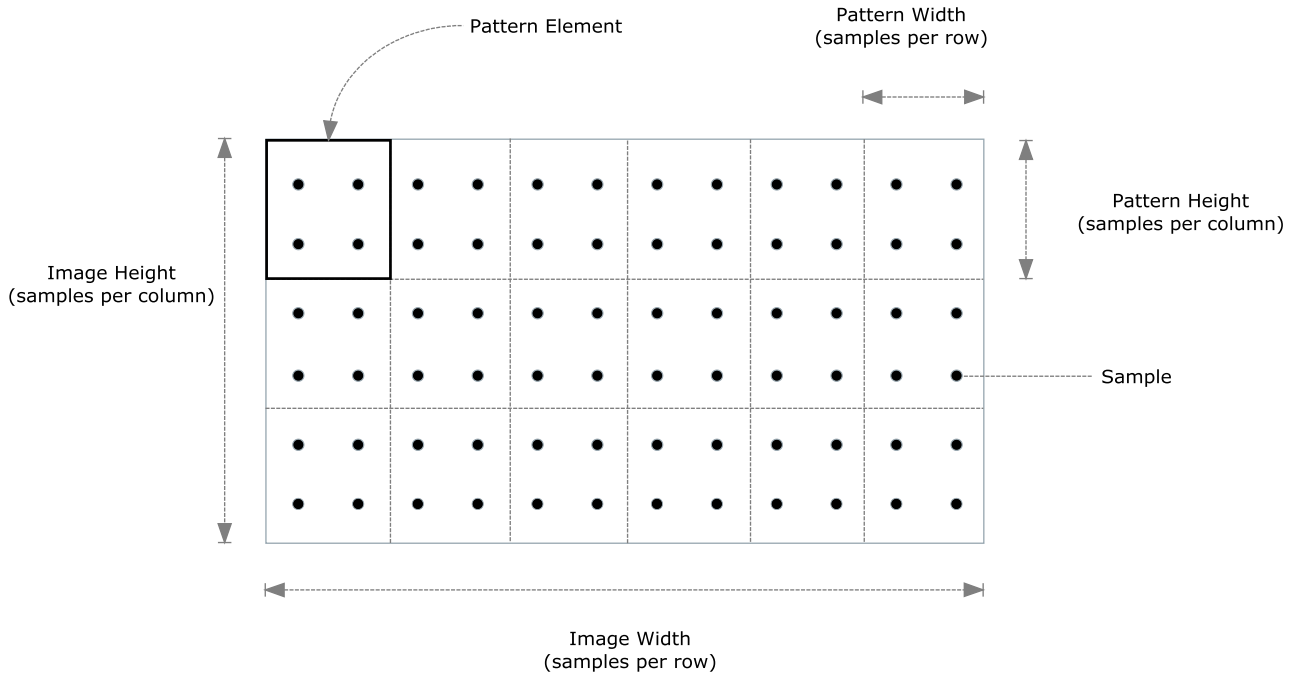

Figure 1 – Sample array

Table 6 – File formats used by the test materials

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 must be a valid string format specification for inserting the channel number (Section 7.3).
dpx	Image file compliant with SMPTE ST 268 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).

With the exception of the DPX file format, all file formats listed in Table 6 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 must be 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.

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)

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 scripts directory.

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 provided 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:

```
boxes/1920x1080/b64a/boxes-1920x1080-0000.b64a

boxes/1920x1080/byr4/boxes-1920x1080-0000.byr4

boxes/1280x720/b64a/boxes-1280x720-0000.b64a

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

```
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 for the testcase. The subdirectory path is formed from 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:

```
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 7. For example, to run the boxes and gradient test cases in the media directory installed along with the test materials using the release configuration of the sample encoder and reference decoder built using the Visual Studio 2005 compiler:

```
testcodec.tcl –m ../media –b release –c VS2005 –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.

Table 7 – Tcl test script command-line options

Option	Description
-r RootDir	Top-level directory for the installation of the VC-5 test materials. (The default depends on the platform)
-m MediaDir	Top-level directory that contains the test cases. (The default depends on the platform)
-b Configuration	Release or debug build configuration. (The default build configuration is release)
-c compiler	Version of the Visual Studio compiler used to build the test materials encoded as “VS” followed by the four digit year for the version of the compiler (example: VS2005). Windows operating system only.
-t TestCases	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	Parts option passed to the codec as a command-line parameter.

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.1.
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 A.2.
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,
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 the Test Materials

The test materials are distributed as a zip file that contains the source code, build and test scripts, sample images, and reference bitstreams that are compliant with the VC-5 essence standard. The sample images can be used to verify that the VC-5 sample encoder is working correctly. The reference bitstreams can be used to verify that the VC-5 reference decoder is working correctly and to test the conformance of VC-5 decoder implementations to the VC-5 essence standard.

Unzip the VC-5 distribution into an empty directory.

The distribution includes the following folders:

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

encoder
Source code for the sample encoder.

decoder
Source code for the reference decoder.

tables
Codebook used by the sample encoder and reference decoder.

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

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

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.

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

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

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 top-level directory of the test materials distribution.

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

Annex A Reference Bitstreams and Images

A.1 Reference Bitstreams

Bitstream Files

The test materials include the VC-5 bitstreams listed in Annex A.1. 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 bitstreams are located in the master sub-directory in each test case organized by VC-5 part number and build configuration.

VC-5 Part 1 Bitstreams

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

VC-5 Part 3 Bitstreams

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

VC-5 Part 4 Bitstreams

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

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

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

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

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

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

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

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

A.2 Reference Images

Image Files

The test materials include the following images that were decoded using a release build of the reference decoder from the reference bitstreams (Annex A.1).

The decoded images are located in the master sub-directory in each test case organized by VC-5 part number and build configuration.

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

VC-5 Part 1 Images

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

VC-5 Part 3 Images

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

VC-5 Part 4 Images

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

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

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

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

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

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

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

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

Annex B Bibliography (Informative)

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

SMPTE ST 377-1:2011, Material Exchange Format (MXF) — File Format Specification, Annex G

ISO/IEC 13818-2:2013 Information Technology — Generic Coding of Moving Pictures and Associated Audio Information — Part 2: Video

Samuel P Harbison and Guy L. Steele, C: A Reference Manual (5th Edition), McGraw-Hill, 2002

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