# VESA Display Stream Compression v1.2a

## Conformance Test Guideline (DSC CTG)

**Revision 1.1**

**1 November, 2019**

## Purpose

The purpose of this document is to provide a test guideline for the VESA® Display Stream Compression (DSC) Standard that can be used by adopters to incorporate into their own test plans.

## Summary

The Display Stream Configuration Conformance Test Guideline (DSC CTG; *VESA DSC 1.2a CTG*) is a recommended test flow and accompanying software for testing encoder or decoder implementations of the VESA DSC Standard (*DSC v1.2a*). The package contains this Guideline, software scripts, associated files, and recommended pictures.

These testing recommendations are not specific to any particular transport and should be incorporated into a test suite that comprehends transport of bits from a Source device to a Sink device.

# Contents

# Tables

# Figures

# Preface

## Intellectual Property

## Trademarks

## Support for this Guideline

Clarifications and application notes to support this Guideline may have been written. To obtain the latest Guideline and any supporting documentation, contact VESA.

If you have a product that incorporates Display Stream Compression (DSC), ask the company that manufactured your product for assistance. If you are a manufacturer, VESA can assist you with any clarification you might need.

Submit all comments or reported errors to support@vesa.org.

## Acknowledgments

This Guideline would not have been possible without the efforts of VESA's DSC Task Group. In particular, Table 1 lists the individuals and their companies that contributed significant time and knowledge to this revision of the Guideline.

**Table 1: Main Contributors to *DSC 1.2a CTG r1.1***

| Company | Name | Contribution |
|---|---|---|
| Advanced Micro Devices, Inc. | Jim Hunkins | Author, Co-chair DSC Task Group |
| Avatar Tech Pubs | Trish McDermott | Technical Writer/Editor |
| Bitec Spain SL | Marco Denicolai | Technical Content |
| Broadcom Inc. | Fred Walls | Author, Technical Content |
| Hardent | Simon Bussières | Technical Content |
| Samsung Electronics Co., Ltd. | Dale Stolitzka | Reviewer, Chair DSC Task Group |

## Revision History

**Table 2: Revision History**

| Date | Revision | Description |
|---|---|---|
| November 1, 2019 | 1.1 | • Modified to support *DSC v1.2a*<br>• New test image and improved Buffer stress image<br>• Removed outdated DSC CTG Buffer Conformance Tool section and associated entries<br>• Updated to current VESA style guidelines<br>• Applied editorial fixes |
| April 27, 2015 | 1.0 | Initial release of the Guideline. |

# 1 Introduction

## 1.1 Scope

The intended audience of this Guideline is DSC encoder and decoder designers and implementers, transport specification writers, and test coverage engineers.

This Guideline applies to the DSC functions and related bitstreams. The transport layer that conveys the DSC bitstream between the encoder and decoder implementations is beyond the scope of this Guideline. It is assumed that testing for the transport layer is documented elsewhere.

Some modes of the VESA® DSC Standard (*DSC v1.2a*) are not commonly implemented, and therefore are **not** directly documented within the scope of this Guideline. For example, all testing within this Guideline assumes a Constant Bit Rate (CBR) mode and does **not** apply to DSC's Variable Bit Rate (VBR) mode. It is a straightforward exercise to extend the tests defined within this Guideline to document these additional modes if needed.

## 1.2 Document Organization

This Guideline is organized as follows:

- Section 1 – Introduction

  This section provides a basic explanation of what this Guideline documents and its expected readership. This section also includes a glossary of terms and references.
- Section 2 – Overview and Section 3 – CTG System Conformance Validation Model

  These sections describe a workflow to verify operation of an implementation under test, using specified pictures and bitstreams.
- Section 4 – Tests

  This section describes specific tests that use the DSC software model for verifying different aspects of the encoder and decoder implementations.
- Appendix A – Installing Software Needed for Tools

  This appendix provides an installation guide for the software that is used by the tests.
- Appendix B – Usage Notes for Picture Sizing

  This appendix provides notes for picture sizing.
- Appendix C – CTG File Summary

  This appendix summarizes the files and formats that are used by the testing tool.
- Appendix D – DSC Encoder and Decoder Picture Coverage Information

  This appendix summarizes the coverages by each suggested picture.
- Appendix E – Test Pictures

  This appendix provides information related to all included test pictures and how to generate test pictures for a specific configuration.

- Appendix F – Using .cfg Files with Test Pictures

  This appendix includes a sample .cfg file and information related to using the file with different package tests.
- Appendix G – Main Contributor History (Previous Revisions)

## 1.3 Acronyms and Abbreviations

Table 1-1 defines acronyms and abbreviations that are used throughout this Guideline. Capping is applied in the definition to indicate the letters used in acronyms and abbreviations.

**Table 1-1: Acronyms and Abbreviations**

| Term | Definition |
|---|---|
| BP | Block Prediction. |
| bpc | bits per component. |
| bpp | bits per pixel. |
| CBR | Constant Bit Rate. |
| .cfg | DSC software model configuration file name extension. |
| CRC | Cyclical Redundancy Check. |
| CTG | Conformance Test Guideline. |
| .dpx | *SMPTE ST 268* file name extension. |
| .dsc | display stream compression file name extension. |
| DSC | Display Stream Compression (VESA). |
| ICH | Indexed Color History. |
| MPP | MidPoint Prediction. |
| .ppm | portable pixel map file name extension. |
| PPS | Picture Parameter Set. |
| QP | Quantization Parameter. |
| rbs | rate buffer size. |
| RC | Rate Control. |
| VBR | Variable Bit Rate. |
| VESA | Video Electronics Standards Association. |
| VLC | Variable Length Code. |
| .yuv | Raw YCbCr pixel data file name extension. The samples are interleaved (CbYCrY) for 4:2:2 pixel data and planar (YCbCr) for 4:2:0 pixel data. |

## 1.4 Glossary

Table 1-2 defines terms that are used throughout this Guideline.

**Table 1-2: Glossary of Terms**

| Term | Definition |
|---|---|
| 4:2:0 | Format for YCbCr video in which the chrominance components are horizontally and vertically subsampled by 2. |
| 4:2:2 | Format for YCbCr video in which the chrominance components are horizontally subsampled by 2. |
| 4:4:4 | Format for RGB or YCbCr video in which the chrominance components are **not** subsampled. |
| bits per component | bpc. Number of bits for each component sample encoded in the bitstream. |
| bits per pixel | bpp. Number of bits sent from an encoder and received by a decoder per unit of pixel time. The bits per pixel rate can have a non-integer value, in which case the number of bits received averaged over a number of successive pixels is an integer. |
| bitstream | Compressed pixel data stream of bits that conform to *DSC v1.2a*. Represents the effects of the multiplexing functions specified by this Guideline, as well as the various layers. |
| Buffer_stress | Test type designed to stress buffer flow controls that prevent encoder/decoder buffer overflow and underflow. |
| checksum | Numerically calculated value in 32 bits that verifies a bitstream, uncompressed source picture, or reconstructed picture. |
| decoder | Implementation of *DSC v1.2a* that reads a bitstream and generates a reconstructed picture. |
| encoder | Implementation of *DSC v1.2a* that reads an uncompressed pixel stream or data file and generates a bitstream. |
| layer | Portion of the *DSC v1.2a* hierarchy. A DSC bitstream can differ from a combination of bits from different layers due to the actions of the multiplexing functions specified within *DSC v1.2a*. |
| picture | Single frame (or interlaced field) of pixels. Also referred to as an "image" within this Guideline. |
| Picture Parameter Set | PPS. Set of parameters that is **optionally** transmitted at the start of a coded picture that provides information that is needed to decode the picture. |
| reconstructed picture | Picture created as DSC decoder output. |
| sample | One component (one of Y, Co, or Cg for RGB input, –or– one of Y, Cb, or Cr for YCbCr input) of one pixel. |
| Sink device | Functional block that contains at least one *DSC v1.2a* decoder implementation and reconstructed pixel stream output. |
| slice | Independently decodable set of compressed bits that represents a specified set of samples. The set of samples forms a rectangle in the horizontal and vertical dimensions. Decoding of any one slice does **not** depend on the availability or decoded result of another slice. |
| Source device | Functional block that contains at least one *DSC v1.2a* encoder implementation and picture source or uncompressed pixel stream to an encoder. |
| uncompressed source picture | Picture used as DSC encoder input. |
| transport | Standards that define their own transport method that uses DSC. |
| visually lossless | Difference between an uncompressed source picture or picture sequence and the same picture or picture sequence after compression and reconstruction that is **not** detectable by the human eye. |

## 1.5 Conventions

### 1.5.1 Precedence

If there is a conflict between text, figures, and tables, the precedence shall be tables, figures, and then text.

### 1.5.2 Keywords

Table 1-3 lists keywords that differentiate between the levels of specifications and options within this Guideline.

**Table 1-3: Keywords**

| Keyword | Definition |
|---|---|
| Informative | Describes information that discusses and clarifies features and mandatory specifications as opposed to mandating them. |
| May | Indicates a choice with no implied preference. |
| N/A | Indicates that a field or value is **not** applicable and has no defined value, and shall **not** be checked or used by the recipient. |
| Normative | Describes features that are specified by this Guideline. |
| **Optional** | Describes features that are **not** specified by this Guideline. However, if an **optional** feature is implemented, the feature shall be implemented as defined by this Guideline (**optional** normative). |
| Shall | Indicates a mandatory specification. Designers are to implement all such specifications to ensure interoperability with other compliant devices. |
| Should | Indicates flexibility of choice with a preferred alternative. Equivalent to the phrase “it is recommended that.” |

## 1.6 Reference Documents

**Table 1-4: Reference Documents**

| Document | Version/ Revision | Date | Referenced As |
|---|---|---|---|
| *MIPI Display Serial Interface 2 Specification*[a] | Version 1.0 | November 17, 2015 | *MIPI DSI-2* |
| *SMPTE ST 268-1:2014 - SMPTE Standard - File Format for Digital Moving-Picture Exchange (DPX)*[b] | Version 2.0 | November 1, 2014 | *SMPTE ST 268* |
| *VESA Glossary of Terms*[c] | Current | Current | |
| *VESA Display Stream Compression (DSC) Standard*[c] | Version 1.2a[d] | January 18, 2017 | *DSC v1.2a* |
| *VESA DisplayPort (DP) Standard*[c] | Version 2.0[d] | June 26, 2019 | *DP Standard* |
| *VESA DSC 1.2a DSC Tools Application Note*[c] | Version 1.2 | January 16, 2018 | *DSC Tools Application Note* |
| VESA *DSC C Model*[c] | Version 1.57 | December 12, 2016 | |
| *VESA DSCParameterValuesVESA spreadsheet*[c] | Version 1.2 | June 21, 2017 | *DSCParameterValues spreadsheet* |
| *DSC 1_2 CTG Coverage Main.xlsx* | Version 1.2 | 2019 | *DSC 1_2 CTG Coverage Main.xlsx* |
| *VESA Intellectual Property Rights (IPR) Policy*[c] | 200D | March 27, 2017 | |

*a. See mipi.org.*
*b. See ieeexplore.ieee.org/Xplore/.*
*c. See vesa.org/vesa-member-downloads/.*
*d. This version of the referenced Standard/Specification is correct at the time of publication of this Guideline. In the event that a later version of the referenced Standard/Specification is published, reference should be made to the latest published version.*

Additional published notes may be made available at future times for additional guidance in using the DSC software model, the DSC CTG, and/or different display standard transports related to DSC. It is up to the reader to check the appropriate locations for such additional documentation.

# 2 Overview

The CTG package contains this Guideline, software scripts, associated files, and recommended pictures to be used when testing a VESA DSC encoder or decoder to determine *DSC v1.2a* compliance. *DSC v1.2a* specifies a coding system and a Picture Parameter Set (PPS) that conveys a set of parameters that is necessary to encode a picture and decode the compressed pixel stream. Specific implementations may be tested within the context of a specific transport layer using the guidelines and tools provided within this package. *DSC v1.2a* specifies encoding and decoding processes that exactly determine the output of the signal chain stages illustrated in Figure 2-1 and described in Table 2-1. Stage 2 encodes a given picture represented as an uncompressed pixel stream that uses normative *DSC v1.2a* C code with a given PPS. The encoding's output yields an identical bitstream at Stage 2's output. Likewise, Stage 5 decodes any received bitstream into an identical reconstructed pixel stream in Stage 6. The deterministic nature of *DSC v1.2a* permits the identical comparison of the observable bitstream that is transported between Stages 3 and 4, with the encoded representations and bit-exact comparison of the reconstructed pixel stream with the decoded picture representations documented in the package.

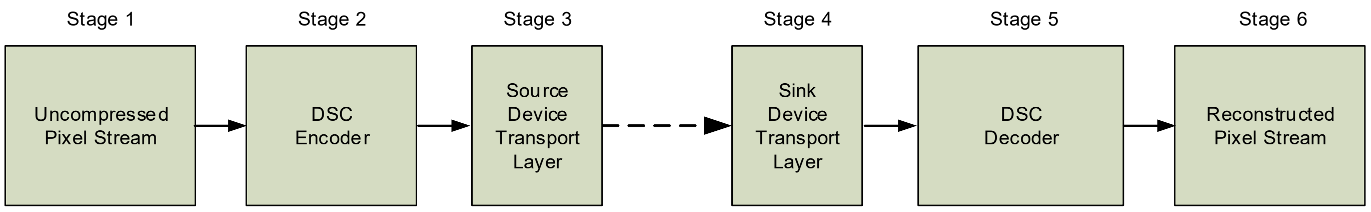
Stage 1
Stage 2
Stage 3
Stage 4
Stage 5
Stage 6
Uncompressed Pixel Stream
DSC Encoder
Source Device Transport Layer
Sink Device Transport Layer
DSC Decoder
Reconstructed Pixel Stream

**Figure 2-1: Typical Display Data Flow**

**Table 2-1: Typical Display Data Flow Stages**

| Stage | | Description |
|---|---|---|
| **#** | **Name** | |
| 1 | Uncompressed Pixel Stream | Picture to be encoded is represented in a .dpx, .ppm, or .yuv file format when used in conjunction with the DSC software model. |
| 2 | VESA DSC Encoder | Encoder compresses the full pixel display into a fixed-rate compressed bitstream. |
| 3 | Source Device Transport Layer | Includes all components and protocols that allow a Source device to transport the bitstream from a DSC encoder. This portion of the transport layer is beyond the scope of this Guideline. *DSC v1.2a* includes an output rate buffer to convert the entropy encoder's variable rate to the link's fixed rate. This rate buffer **cannot** overflow or underflow. |
| 4 | Sink Device Transport Layer | Includes all components and protocols that allow a Sink device to receive a bitstream by way of a DSC decoder. All portions of the transport layer are beyond the scope of this Guideline. *DSC v1.2a* includes an input rate buffer as part of the Standard to convert the link's fixed rate to the entropy decoder's variable rate. This rate buffer **cannot** overflow or underflow. |
| 5 | VESA DSC Decoder | Decodes the compressed bitstream, which results in the reconstructed picture. This reconstructed picture is deterministic for a given compressed bitstream. |
| 6 | Reconstructed Pixel Stream | Picture-receiving unit may display the reconstructed pixel stream. DSC software model shall output the reconstructed picture in a .dpx, .ppm, or .yuv file format. |

# 3 CTG System Conformance Validation Model

Display end-to-end system interoperability testing is performed during simulation, using the VESA DSC software model. Figure 3-1 illustrates and Table 3-1 describes a typical system conformance validation flow that uses cyclic redundancy checks (CRCs) at critical stages. This same process is followed regardless of the different configurations suggested in this Guideline.

The CRC calculation method (algorithm and mapping) shall match that of the transport standard that implements DSC. For example, a DisplayPort display uses the 16-bit CRC calculation explained in *DP Standard, Appendix J*.

The CTG package uses a *DSC v1.2a* CRC value to compare the uncompressed pixel stream, bitstream, and reconstructed pixel stream with a device under test. A software tool is included that generates CRC values for use by VESA DisplayPort (DP) or MIPI® implementations.
If using a hardware-generated CRC that is different, the tester shall generate the needed CRCs, as necessary. The CRC should be run only on the raw pixel data (i.e., uncompressed, compressed, or reconstructed), excluding any file headers or transport-added additional structures.

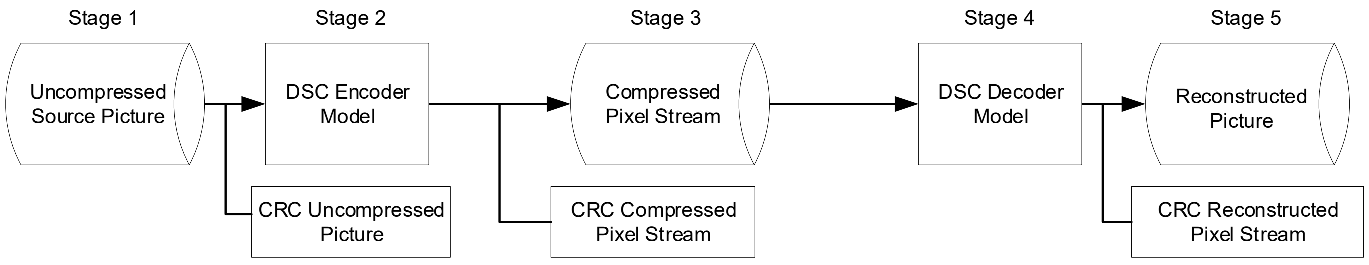

**Figure 3-1: Conformance Guideline Validation Flow**

**Table 3-1: Conformance Guideline Validation Flow Stages**

| Stage | | Description |
|---|---|---|
| **#** | **Name** | |
| 1 | Uncompressed Source Picture | Source picture file is uncompressed display data that should match the intended final display resolution and other parameters. Multiple passes through the validation flow are used with different source picture files. Picture file contents should be selected to stress the encoder and decoder functional paths and the rate buffer overflow and underflow limits.<br>Some transport implementations may have further transport buffer considerations in addition to the DSC buffers. In those cases, the pictures could be designed to stress both the DSC and transport buffers working together. |
| 2 | DSC Encoder Model | Encoder can be tested independently with the source picture data files and related CRC for the uncompressed source picture and compressed bitstream.<br>Uncompressed source picture file from Stage 1 is compressed using a PPS that is defined to match the Stage 3 transport model's transport rate. The compressed output bitstream should match that of the executable DSC encoder model.<br>Each time the same source picture is run through the encoder with the same PPS values, the encoder should always produce the same CRC at the end of Stage 2. |
| 3 | Compressed Data Transport Layer | Compressed bitstream produced in Stage 2 should be transported through the display standard under test's transport layer. This test model assumes that data corruption will **not** occur during the transport from Stage 2. Input data to Stage 3 should exactly match the transport layer's output data. Any transport-added pieces need to be removed prior to being submitted to the DSC Decoder Model. CRC value does **not** change when the compressed data passes through Stage 3. |
| 4 | DSC Decoder Model | Decoder can be tested in the same manner as the encoder, but with the compressed bitstream as the Source device, and the compressed bitstream and reconstructed pixel stream CRCs.<br>The VESA DSC decoder model is used at this stage to reconstruct the pixel data and provide a visually lossless picture. The reconstructed pixel stream from the decoder under test shall match the reconstructed picture created by the VESA DSC software model. |
| 5 | Reconstructed Picture Display | As expected, the uncompressed source picture will **not** match bit-for-bit to the reconstructed picture because the picture has been lossy compressed.<br>If the same compressed bitstream is decoded more than once, the reconstructed picture should have the same CRC value for each run because the decoder algorithm is deterministic. |

# 4 Tests

This Guideline provides three sets of tests:

- DSC Algorithm Core Coverage
- Slice Isolation Coverage
- Buffer Control Coverage

For the DSC Algorithm Core Coverage and DSC Algorithm Core Coverage tests, a picture at a specific resolution is provided. This picture was specifically chosen and modified to exercise a large portion of the DSC encoder and decoder logic.

For the Buffer Control Coverage test, a script is provided to generate an image to specifically stress the rate buffer implementation.

## 4.1 Picture-specific Tests

The pictures and tests that are documented in this Guideline are designed to use the recommended PPS values, as suggested in *DSC Tools Application Note* and *DSCParameterValues spreadsheet*. Designers who choose to implement non-recommended values should determine whether coverage is sufficient as is, or if they need to expand the test cases beyond those included in this Guideline.

The test coverage shown is derived with Variable Bit Rate (VBR) mode disabled. At this time, if an implementer chooses to enable VBR mode, they should also review the coverage in that mode and add coverage as needed.

The test picture is provided as 1080P-sized RGB 4:4:4 .dpx files in 8, 10, 12, and 16 bits per component (bpc). It is up to the test operator to convert to larger sizes or to other color modes, such as 4:4:4 YUV, 4:2:2, or 4:2:0, as needed.

*DSC Tools Application Note* or *DSCParameterValues spreadsheet* are used to create the PPS for each test. If the images are resized, a new PPS for each should be generated based on each image's new size and slice configuration. The PPS used will impact the CRC results. Therefore, the pictures should be run through the software models to generate the correct CRCs to match the desired PPS case(s). CRCs are used at each of the following points:

1 Uncompressed source picture (Stage 1 feeding into Stage 2).

2 Compressed pixel stream (end of Stage 2, input to Stage 4).

3 Final reconstructed picture (end of Stage 4).

See Section 4.3 for recommended image resizing and conversion methods.

## 4.2 Running Tests

Because DSC encoders and decoders are expected to always provide deterministic outputs that match the DSC software model output, testing can be done with a few different approaches, as described in the following subsections.

### 4.2.1 Encoder

Testing the encoder requires a method to capture its compressed output bitstream. The bitstream could then be compared to a known correct compressed output bitstream generated from the same uncompressed source picture and parameters. This good test bitstream can be generated either from the DSC software model or from a known good encoder that is DSC-compliant.

Alternatively, the compressed bitstream could be sent through a decoder (either the DSC software model or a known good decoder) and the reconstructed output could be compared with the known good reconstructed picture from either the DSC software model or a known good encoder/decoder hardware implementation.

The DSC software model outputs a .dsc file with the compressed bitstream for the uncompressed source file. This same file is then used by the DSC software model decoder to reconstruct the output picture.

### 4.2.2 Decoder

Testing the decoder requires a compressed bitstream that has been previously generated, either from the DSC software model or from a known good encoder that uses the same uncompressed source picture and parameters. The reconstructed output from the decoder is expected to match the reference reconstructed output from another good decoder that is DSC-compliant or the DSC software model for the decoder. Some graphics devices apply additional adjustments to the output picture between the frame buffer and the display. In those cases, the unmodified picture that is received directly from the frame buffer might be needed in the comparison (i.e., the decoder's direct output).

A common way to verify that the compressed bitstream (for the encoder) or reconstructed picture (for the decoder) match the test references is to calculate a CRC value for the test reference(s) and generated output(s). VESA supplies a software tool to generate CRCs that is based on *DP Standard* and *MIPI DSI-2*. However, any CRC standard method can be used as long as it is consistent for both the device under test and the test reference data.

It is important to understand that the DSC software model does **not** include any modeling of the actual transport. The transport might map the bit ordering within the stream in a specific way, include additional control bytes, etc. Therefore, one of the following should be done:

- Capture the compressed bitstream before or after the transport is involved (i.e., before the transport receives it or after the compressed bitstream is recovered from the transport as it enters the decoder), –or–
- Enhance the DSC software model to also model the transport in use.

## 4.3 DSC Algorithm Core Coverage

Running the DSC1_2-Test_Image_nnbpc.dpx picture at the highest supported input bpc and in 4:4:4 RGB color format provides the DSC algorithm core test coverage. Running the same image at the same input bpc but in one of the YUV color formats adds additional coverage for the `VLC_PATHS` and `RC_RANGES` parameters. The available *DSC v1.2a*-supported YUV color formats are listed below in order of the highest additional coverage:

- 4:2:2
- 4:4:4
- 4:2:0

Not all formats will be available for all transports. Running the test image in additional formats will **not** necessarily provide additional DSC algorithm core coverage. The transport tests can include additional runs to test different features, formats, or other options. If run at the supplied 1920x1080 at 60Hz resolution, the pictures should be run with slice dimensions of 1920x108.

For designs that support resolutions in which having 1 slice/line fits the design's supported maximum bandwidth, run the tests with the pictures at their original sizes. For the higher resolutions and/or refresh rates that need more than 1 slice/line, see Section 4.3.1.

For the DSC core algorithm logic, changing the picture resolution and/or orientation has minimal effect on test coverage. Therefore, running the DSC1_2-Test_Image_nnbpc.dpx picture as described provides high coverage. See Appendix D for core test coverage information for each picture variation. See Section E.1 for a representation of the test image.

### 4.3.1 Multiple Slices/Line Coverage

For implementations that support more than 1 slice/line, tests should be performed using the number of slices/line that is specified by the transport in use. In some cases, different frame rates might result in a different number of slices/line, thus some pictures might need to be tested with different slice/line configurations based on such frame rates.

### 4.3.2 Adjusting Default Picture Size to Fit Specific Display Limitations

While the coverage report is based on the picture resolution and slice sizes provided in this Guideline, using tiling to increase picture sizes should **not** substantially impact test coverage. If the system under test cannot support the supplied resolutions without applying scaling before and after the DSC encoder and decoder, the picture and tile sizes can be adjusted to the resolutions needed by the system. The following guidelines are provided to help maintain test coverage as close to the uncompressed source picture as possible.

1 If the picture is smaller than the display can support:

  a Tile the picture into a grid of pictures, using the original-sized picture for each grid (e.g., one picture becomes a set of four pictures in a 2x2 grid).

  b Crop the newly tiled picture to match the supported resolution.

  The crop should be such that the original picture's upper left corner is also the upper left corner of the newly cropped picture.

2 Adjust the slice size, as needed:

- Based on the slice width, follow the transport's slice width rules for the number of slices/line.
- Keep the number of lines/slice as close as possible to the original 108 lines/slice.
    - Ensure that the lines/slice count follows the transport's rules.

Appendix B provides sample command lines for using ImageMagick® to achieve the above items. Other methods can also be used to achieve the same results. Take care to **not** inadvertently change the bpc or pixel data (e.g., changing the color space from P3 to sRGB can shift pixel values).

## 4.4 Slice Isolation Coverage

If one or more bits become corrupted within a single slice within the compressed stream that is being input to the decoder, the remainder of the same slice will be corrupted in the output picture. In correct operation, the remaining slices will be isolated and not show any corruption. Additionally, the decoder must **not** hang as a result of such corruption.

This CTG package supplies a compressed stream that was originally generated from the DSC1_2-Test_Image_nnpcb.dpx test picture with slice dimensions 1920 pixels × 54 lines. The stream corrupts four separate bytes with single-bit errors – one each in two separate slices and two in a third slice. The resulting picture (after going through the decoder) should visually show that three of the slices are corrupted and have no impact on any other slices. Notes are included on the corrupted decoded example picture file:

- **DSC Compressed Stream File - no errors** – DSC1_2-Test_Image_08bpc - good.dsc
- **DSC Compressed Stream File - 4 bit errors** – DSC1_2-Test_Image_08bpc - 4 corrupted bits.dsc
- **DSC Output File resulting from 4 bit error compressed file** – DSC1_2-Test_Image_08bpc - 4 corrupted bits - out.ppm

It is up to the Transport specifications that reference DSC to include their own tests for the containment of dropped or inserted data into the compressed bitstream if such deletions or insertions are possible.

## 4.5 Buffer Control Coverage

A software script (generate_buffer_image.pl) is provided that algorithmically generates test pictures for the 4:4:4 RGB, 4:2:2 YCbCr, –or– 4:2:0 YCbCr pixel formats. The script and algorithm used are documented in Appendix E. The algorithm is designed so that it can easily be integrated into an encoder design as a hardware test pattern generator for implementations that do **not** allow an arbitrary image to be input bit-accurately. See Appendix D for core test coverage information for the buffer control coverage.

# A Installing Software Needed for Tools

The following software/utilities are used by the tools described in this Guideline. Installation and operation vary between systems; therefore, be sure to check and follow installation instructions related to the OS in use. To help minimize the test system debug, each tool should be tested and confirmed to be operational before attempting to run any of the software scripts identified in this CTG package.

## A.1 Installing ImageMagick

ImageMagick can be downloaded and installed from imagemagick.org. See Appendix B for picture sizing usage notes.

## A.2 Installing MATLAB to Run Down Sampling MATLAB Scripts

A MATLAB® source code file, included with the CTG files, provides the ability to down sample the supplied 4:4:4 RGB images to the 4:2:2 YUV –or– 4:2:0 YUV format. To run these files, either MATLAB or GNU Octave should be used.

See the MathWorks website's MATLAB section for information on how to obtain and install MATLAB at mathworks.com/products/matlab.html?s_tid=hp_products_matlab.

## A.3 Installing GNU Octave for Running MATLAB-based Down Sampling Scripts

GNU Octave can be installed on multiple platforms. Installation information is available at gnu.org/software/octave/.

# B Usage Notes for Picture Sizing

Although larger and smaller images can be used to test different resolutions, test coverage for resolutions other than 1920x1080 might vary. Therefore, when possible, each implementation should be tested using an image size of 1920x1080 (in addition to the other supported sizes) to ensure adequate coverage.

If other image sizes are needed, the following guidance is supplied.

## B.1 Creating a Test Image That Has the Same or Larger Dimensions as the Primary Test Image

To make an image larger and maintain DSC engine test coverage, without rescaling the image, the following procedure should be used.

1 Enlarge the image by tiling it to a size that is greater than or equal to the final size needed.

2 If necessary, crop the larger image to the final size needed, with the upper left corner of the final image matching that of the original image.

The following example shows one possible method of using ImageMagick. Other programs can be used to accomplish the same effect. The 1920x1080 DSC1_2-Test_Image.ppm picture is used as the source image in the example.

```
montage DSC1_2-Test_Image.ppm DSC1_2-Test_Image.ppm DSC1_2-Test_Image.ppm DSC1_2-Test_Image.ppm
-geometry 1920x1080 -tile 2x2 montage_3840x2160.ppm
```

```
convert montage_3840x2160.ppm -crop 2560x1600 FinalFile_2560x1600.ppm
```

***Note:*** *For further details regarding ImageMagick command usage, see the documentation included with ImageMagick.*

## B.2 Creating a Test Image That Has at Least One Smaller Dimension than the Primary Test Image

Some users might need images that are smaller than 1920x1080 for testing lower-resolution modes. The following describes a process that can be used to create lower-resolution test images. Figure B-1 illustrates how this process can work for creating nine 640x480 test images from a test image that was originally 1920x1080.

1 If the original image's width and height are not both integer multiples of the smaller image dimensions needed, do the following:

   a Use tiling to create a larger image.

   b If the resulting image does not result in integer multiples of the smaller image dimensions needed, crop the tiled image to have the needed width and height.

2 Split the tiled image into individual, non-overlapping cropped images that can each be encoded and decoded.

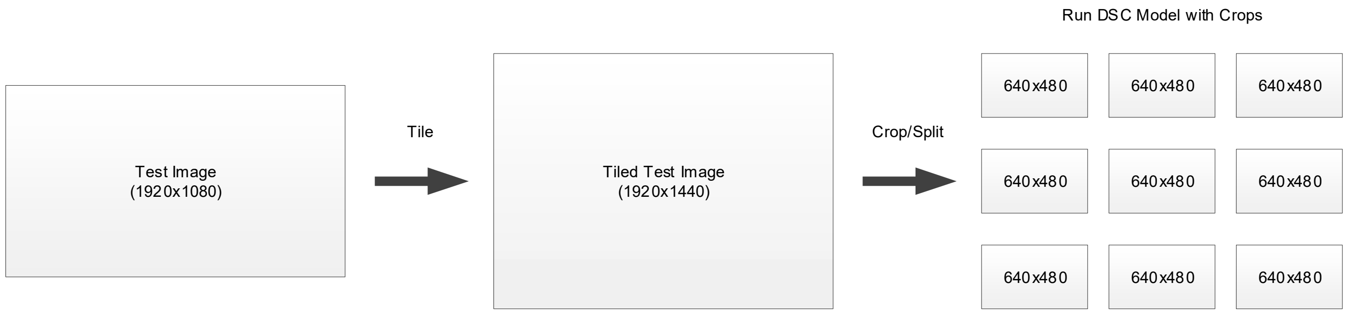

**Figure B-1: Example Process for Creating Nine 640x480 Test Images from a Single 1920x1080 Test Image**

The following example shows one possible method of using ImageMagick to do what is illustrated in Figure B-1.

```
montage example.png example.png -geometry 1920x1080 -tile 1x2 example_1920x2160.png
convert example_1920x2160.png -crop 1920x1440 example_1920x1440.png
convert example_1920x1440-0.png -crop 640x480 example-%02d.png
```

# C CTG File Summary

Table C-1 lists the DSC software model and related files that are included in the software package provided with *DSC v1.2a*.

**Table C-1: DSC C Model Files**

| File Name | Description |
|---|---|
| DSC.exe | Main executable file. |
| README.TXT | Instructions for usage. |
| Source {folder} | Folder that contains the DSC C model source files. |
| test_list.txt | List of pictures to be processed by the DSC code. |
| test.cfg | Main configuration file used by the DSC code. Calls out the DSC baseline configuration files. |

The PPS parameters used with this Guideline come from *DSC Tools Application Note* and are equivalent to the PPS parameters generated by *DSCParameterValues spreadsheet*. To use the *DSC Tools Application Note* method, the C model should be recompiled, following the instructions provided in *DSC Tools Application Note* so that the `GENERATE_RC_PARAMETERS` configuration file option can be set to 1 to automate PPS creation.

Table C-2 and Table C-3 list the files that are included in the CTG package.

Table C-2 lists the test image and C code to generate the Buffer_stress picture that is used in the testing outlined in Section 4.2 and Section 4.4. These files are located in the **Pictures** folder.

**Table C-2: DSC CTG Algorithm Core Coverage Files**

| File Name | Description |
|---|---|
| DSC1_2-Test_Image_nnbpc.dpx | Picture that provides the widest algorithm coverage. Replace nn with the bpc value (i.e., 08, 10, 12, or 16). |
| generate_buffer_image | C code used to create the Buffer_stress picture (see Appendix E). |

Table C-3 lists the compressed stream and corrupted output files used in DSC Slice Isolation Coverage (see Section 4.3). These files are located in the **Slice_Isolation** folder.

**Table C-3: DSC CTG Slice Isolation Files**

| File Name | Description |
|---|---|
| DSC1_2-Test_Image_08bpc - good.dsc | Reference good compressed stream file capture from DSC1_2-Test_Image_08bpc.dpx (slices = 1920 pixels × 54 lines). |
| DSC1_2-Test_Image_08bpc - 4 corrupted bits.dsc | Corrupted compressed stream file capture from DSC1_2-Test_Image_08bpc.dpx (slices = 1920 pixels × 54 lines). |
| DSC1_2-Test_Image_08bpc - 4 corrupted bits - out.ppm | Sample decoded picture from corrupted DSC1_2-Test_Image_08bpc.dpx compressed stream file with notes regarding corruption areas. |

Table C-4 lists a program that will generate the CRC values to be used to verify that the hardware results match those from the DSC software model.

**Table C-4: DSC CTG CRC Generation Program**

| File Name | Description |
|---|---|
| VesaCrcTool | Code that generates the CRC values for *DSC v1.2a* based on either DP or MIPI CRC algorithms. |

Table C-5 lists the compressed MATLAB source code file that should be used to down sample the 4:4:4 RGB images to the 4:2:2 YUV –or– 4:2:0 YUV format (see Section A.2 for further details).

**Table C-5: MATLAB Source Code for Down Sampling 4:4:4 RGB to 4:2:2 YUV –or– 4:2:0 YUV**

| File Name | Description |
|---|---|
| Down_Sample_Source.zip | MATLAB source code to use for down sampling the supplied 4:4:4 RGB images to the 4:2:2 YUV –or– 4:2:0 YUV format. Can be run directly with MATLAB or GNU Octave. |

# D DSC Encoder and Decoder Picture Coverage Information

## D.1 DSC Core Algorithm Coverage

Table D-3 through Table D-5 provide the coverage of different paths through the core algorithm of the pictures shown when run with the slice sizes as noted. The **Maximum Paths** column indicates how many different paths are taken through the specific code section in the software model. **Green** table cells with a ✔ indicate full (100%) coverage of all possible runs through the parameter's code section. **Blue** table cells show improved coverage when adding 4:2:2 YUV mode in addition to 4:4:4 RGB mode. A number in a cell indicates the number of actual paths covered.

Table D-1 lists the PPS settings used for the coverage used in Table D-3 through Table D-5.

**Table D-1: DSC Core Algorithm Coverage PPS Settings**

| Setting | | Value |
|---|---|---|
| Target ***bits_per_pixel*** | 4:4:4 RGB | 8 |
| | 4:2:2 YUV | 7 |
| | 4:4:4 YUV[a] | 8 |
| | 4:2:0 YUV[a] | 6 |
| ***dsc_version_minor*** | | 2 |
| ***block_prediction_enable*** | | 1 |
| Slice Size | | 1920x108 |

a. *The coverage tables that follow show only the two highest coverage modes, 4:4:4 RGB and 4:2:2 YUV. See Section 4.3 for guidance of which modes to run. For further details regarding 4:4:4 and 4:2:0 YUV coverage modes, see the working spreadsheet, DSC 1_2 CTG Coverage Main.xlsx, in **Display Stream Compression Task Group** > **Tools** > **CTG** > **DSC_1_2_CTG** > **DSC 1.2A CTG Files**.*

Due to the algorithm's design, some of the potential paths are **not** expected to run. For example, in the case of the `VBR_UNDERFLOW` parameter, this will happen only if there is a buffer underflow; however, this should be impossible with a correct design when running with VBR disabled. Because the test coverage was generated based on the supplied DSC software model with VBR disabled, that loop was never taken.

Table D-2 lists additional coverage notes with respect to Table D-3 through Table D-5.

**Table D-2: Additional DSC Core Algorithm Coverage Notes**

| Parameter | Note |
|---|---|
| `FLAT_QP_ADJ` | Mix of *DSC v1.1* and *DSC v1.2a* (two runs) that provides coverage of 18 paths (maximum). |
| `N_ADJUSTMENT_BITS` | Can hit only one path per run. To hit other paths, adjusting the `SLICE_WIDTH` and `BITS_PER_PIXEL` parameters may change between the possible nine paths.<br>For example, with 8.0625bpp:<br>• `SLICE_WIDTH` = 940, path = 6<br>• `SLICE_WIDTH` = 950, path = 5<br>• `SLICE_WIDTH` = 960, path = 4<br>• `SLICE_WIDTH` = 980, path = 3 |
| `SCALE` | Not expected to hit higher path values unless varying from the recommended PPS values. |
| `VLC_PATHS` | Paths 7 and 8 are possible only in 16bpc. Path 12 is possible only in Native 4:2:2. |

**Table D-3: Picture Coverage for DSC Core Algorithm – DSC1_2-Test_Image_08bpc: 8bpc**

| Parameter | Maximum Paths | 4:4:4 RGB Only | 4:4:4 RGB and 4:2:2 YUV |
|---|---|---|---|
| QUANT_TABLE_LUMA | 16 | ✔ | ✔ |
| RESIDUAL_SIZES | 9 | ✔ | ✔ |
| PRED_SIZE_ADJ | 100 | 65 | 69 |
| NZ_MIDPOINT | 8 | ✔ | ✔ |
| BP_SEARCH[a] | 8 | ✔ | ✔ |
| ENC_FLATNESS_CHK | 5 | 4 | 4 |
| N_ADJUSTMENT_BITS | 9 | 1 | 1 |
| SCALE | 56 | 25 | 25 |
| RC_RANGES | 15 | 14 | 14 |
| RC_PATHS | 18 | ✔ | ✔ |
| RC_BIT_SAVE | 5 | ✔ | ✔ |
| VLC_PATHS | 13 | 10 | 11 |
| FORCE_MPP | 2 | ✔ | ✔ |
| MPP_BOUNDING | 2 | ✔ | ✔ |
| ICH_USE | 192 | 191 | ✔ |
| FLAT_QP_ADJ[a] | 7 | ✔ | ✔ |
| VBR_UNDERFLOW | 1 | 0 | 0 |
| BP_MODES | 8 | ✔ | ✔ |

a. *When Block Prediction (BP) is disabled in the .cfg files,* `BP_Search` *becomes* `0 – 7` *and* `Flat_QP_Adj` *becomes 1.*

**Table D-4: Picture Coverage for DSC Core Algorithm – DSC1_2-Test_Image_10bpc: 10bpc**

| Parameter | Maximum Paths | 4:4:4 RGB Only | 4:4:4 RGB and 4:2:2 YUV |
|---|---|---|---|
| QUANT_TABLE_LUMA | 20 | ✔ | ✔ |
| RESIDUAL_SIZES | 11 | ✔ | ✔ |
| PRED_SIZE_ADJ | 144 | 85 | 97 |
| NZ_MIDPOINT | 10 | ✔ | ✔ |
| BP_SEARCH[a] | 8 | ✔ | ✔ |
| ENC_FLATNESS_CHK | 5 | 4 | 4 |
| N_ADJUSTMENT_BITS | 9 | 1 | 1 |
| SCALE | 56 | 25 | 25 |
| RC_RANGES | 15 | ✔ | ✔ |
| RC_PATHS | 18 | ✔ | ✔ |
| RC_BIT_SAVE | 5 | ✔ | ✔ |
| VLC_PATHS | 13 | 10 | 11 |
| FORCE_MPP | 2 | ✔ | ✔ |
| MPP_BOUNDING | 2 | ✔ | ✔ |
| ICH_USE | 192 | ✔ | ✔ |
| FLAT_QP_ADJ[a] | 7 | ✔ | ✔ |
| VBR_UNDERFLOW | 1 | 0 | 0 |
| BP_MODES | 8 | ✔ | ✔ |

a. *When Block Prediction (BP) is disabled in the .cfg files,* `BP_Search` *becomes* `0 – 7` *and* `Flat_QP_Adj` *becomes 1.*

**Table D-5: Picture Coverage for DSC Core Algorithm – DSC1_2-Test_Image_12bpc: 12bpc**

| Parameter | Maximum Paths | 4:4:4 RGB Only | 4:4:4 RGB and 4:2:2 YUV |
|---|---|---|---|
| QUANT_TABLE_LUMA | 24 | ✔ | ✔ |
| RESIDUAL_SIZES | 13 | ✔ | ✔ |
| PRED_SIZE_ADJ | 196 | 109 | 132 |
| NZ_MIDPOINT | 12 | ✔ | ✔ |
| BP_SEARCH[a] | 8 | ✔ | ✔ |
| ENC_FLATNESS_CHK | 5 | 4 | 4 |
| N_ADJUSTMENT_BITS | 9 | 1 | 1 |
| SCALE | 56 | 25 | 25 |
| RC_RANGES | 15 | 14 | 14 |
| RC_PATHS | 18 | ✔ | ✔ |
| RC_BIT_SAVE | 5 | ✔ | ✔ |
| VLC_PATHS | 13 | 10 | 11 |
| FORCE_MPP | 2 | ✔ | ✔ |
| MPP_BOUNDING | 2 | ✔ | ✔ |
| ICH_USE | 192 | ✔ | ✔ |
| FLAT_QP_ADJ[a] | 7 | ✔ | ✔ |
| VBR_UNDERFLOW | 1 | 0 | 0 |
| BP_MODES | 8 | ✔ | ✔ |

a. *When Block Prediction (BP) is disabled in the .cfg files, `BP_Search` becomes `0 – 7` and `Flat_QP_Adj` becomes 1.*

## D.2 Buffer Control Coverage

Table D-6 lists the minimum and maximum buffer model levels that the 1920x1080 buffer image hits per slice. The rbsMin value is the maximum that could, in theory, be hit; however, there is a control in the code that prevents the buffer from completely filling to this point. Additionally, a buffer can **never** be allowed to go below 0.

The minimum buffer model fullness per slice is based on the minimum fullness after the ***initial_xmit_delay***. It does **not** include the initial buffer model fullness, which is always 0 per slice (assuming that the DSC and transport buffers are implemented independently; some implementations may combine the two and consideration should be made for the impact of such a design on the testing). Running the image using a higher ***bits_per_pixel*** value will result in improved coverage of the maximum buffer fullness.

**Table D-6: Buffer Model Full/Empty Coverage for Buffer_stress Testing**

| | **DSC1_20-Buffer_stress<br>Slice = 1920 pixels × 108 lines** | |
|---|---|---|
| | **Minimum** | **Maximum** |
| **rbsMin** | | 15744 |
| **Percent Full** | | 99% |
| **For Picture** | 35 | 15586 |
| **Slice #** | | |
| 1 | 95 | 15586 |
| 2 | 35 | 14417 |
| 3 | 35 | 13205 |
| 4 | 35 | 12017 |
| 5 | 35 | 10809 |
| 6 | 35 | 9650 |
| 7 | 35 | 8416 |
| 8 | 35 | 7253 |
| 9 | 35 | 6061 |
| 10 | 35 | 4879 |

# E Test Pictures

## E.1 DSC Algorithm Core Coverage Test Pictures

Table E-1 lists the test pictures that are used for core coverage in Section 4.3 and the resulting test output image for slice isolation in Section 4.4.

**Table E-1: DSC Algorithm Test Picture**

| Picture | Resolution | Coverage Notes | Image |
|---|---|---|---|
| DSC1_2-Test_Image_nnbpc.dpx | 1920x1080 | Widest algorithm path coverage. | Figure E-1 |
| DSC1_2-Test_Image_08bpc - 4 corrupted bits.dpx | 1920x1080 | Shows errors introduced by four 1-bit errors for slice isolation testing. | Figure E-2 |

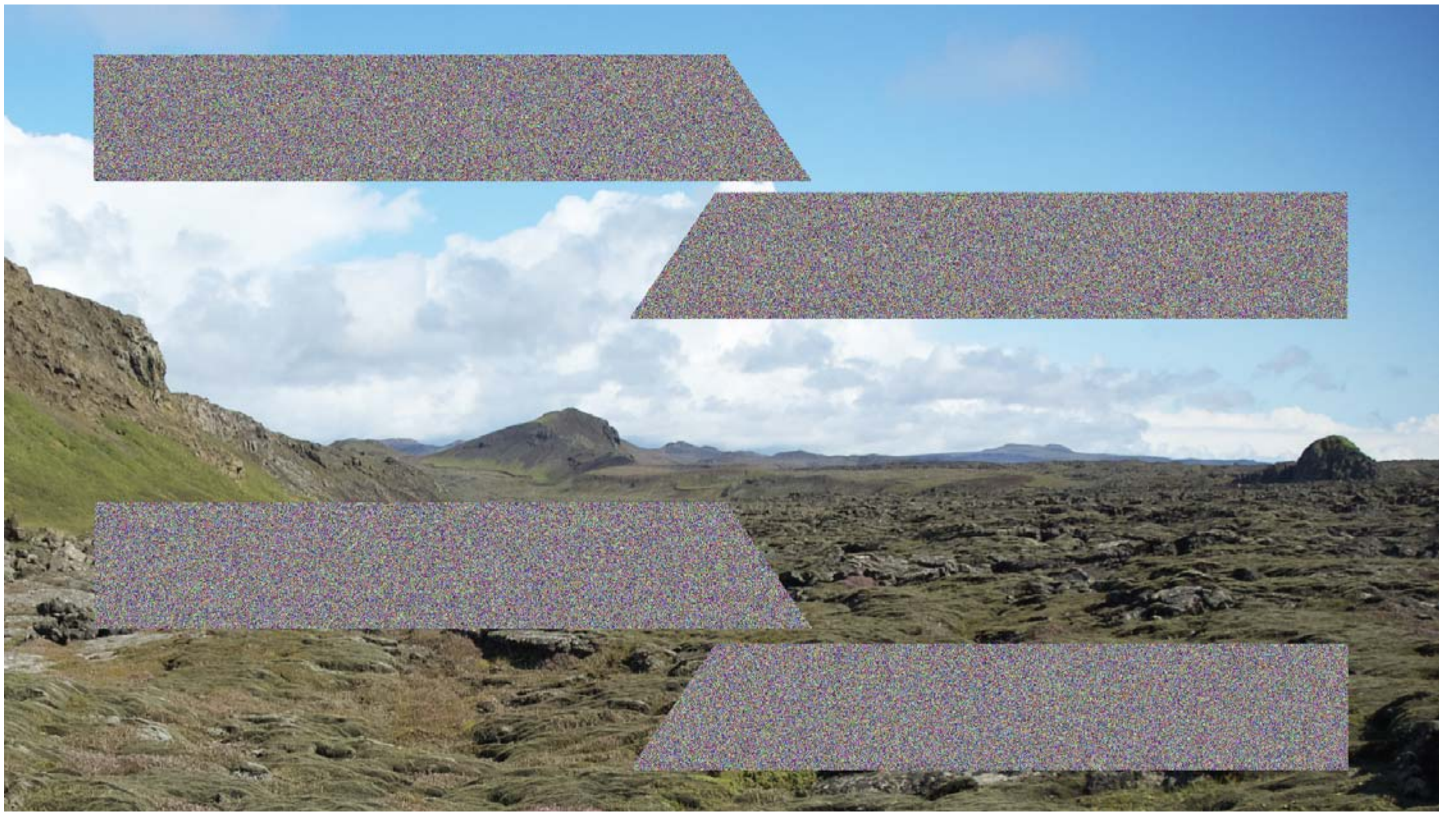

**Figure E-1: DSC1_2-Test_Image_nnbpc.dpx Test Picture Sample**

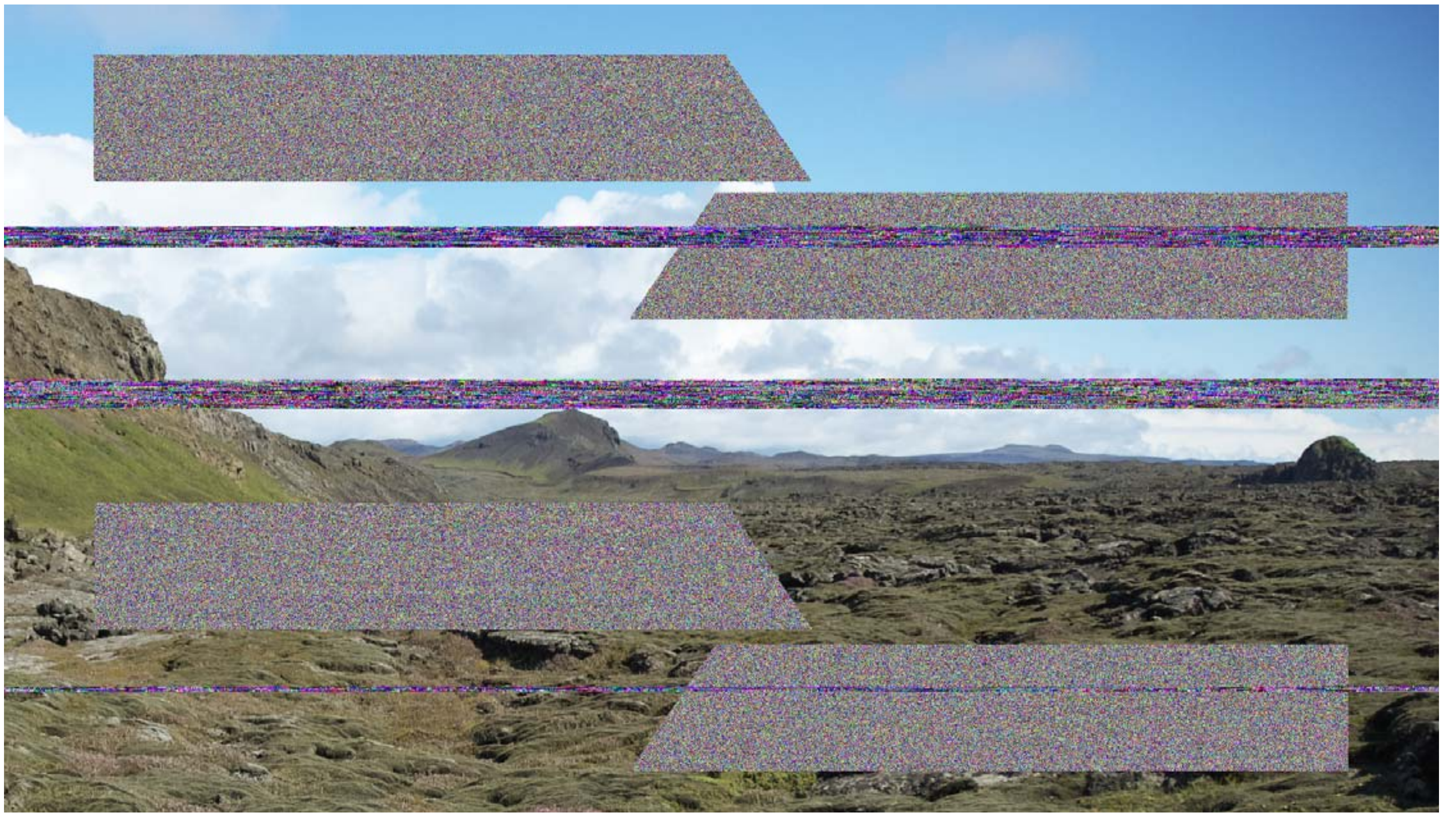

**Figure E-2: DSC1_2-Test_Image_08bpc - 4 corrupted bits.dpx Test Picture Sample**

## E.2 Buffer Control Coverage Test Picture Generation

The CTG package includes code that is used to algorithmically generate a test pattern that provides broad coverage with an emphasis on Buffer_stress. An algorithmically generated test pattern enables a low level of verification in some scenarios (e.g., certain types of scalers or repeaters) where bit-accurately encoding the core image might not be possible. The test pattern allows for a straightforward hardware implementation that uses approximately 5000 gates (see Figure E-3).

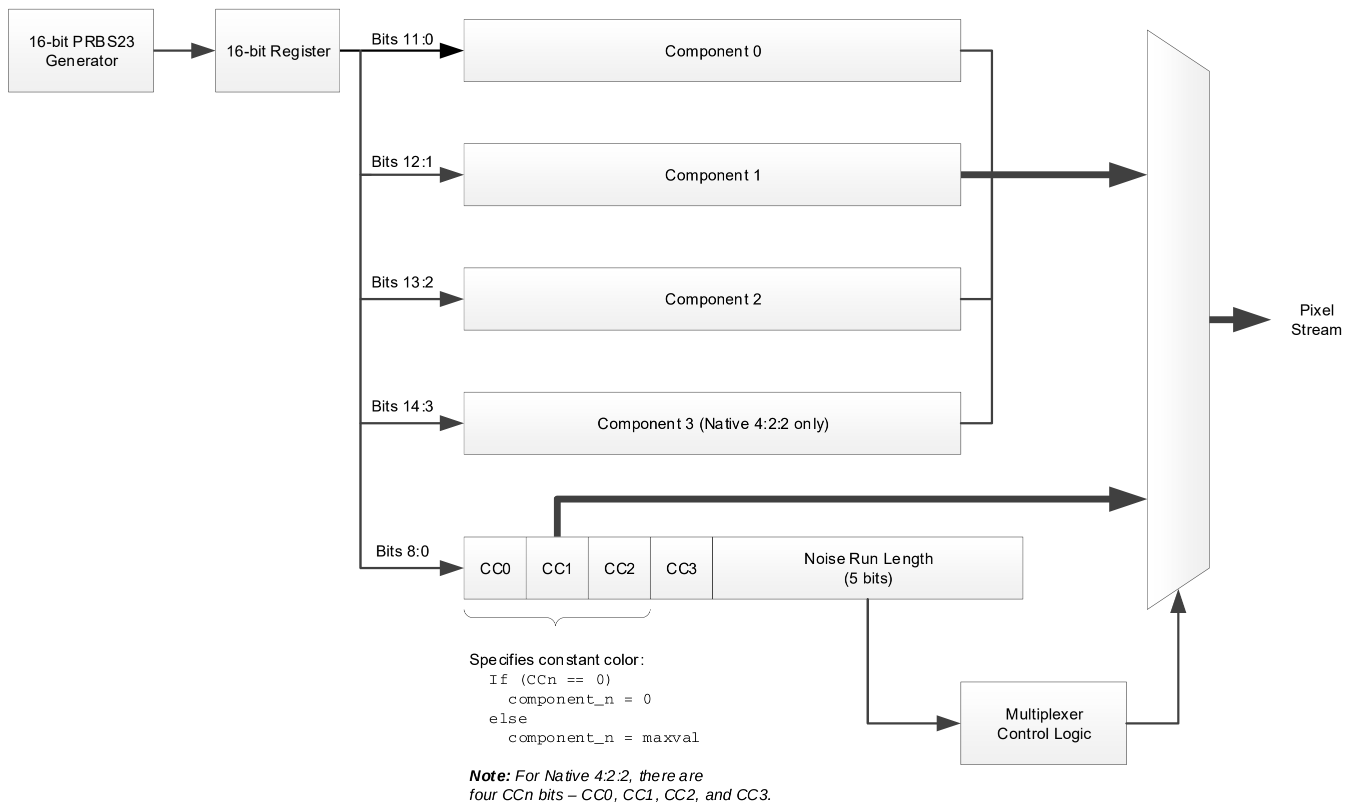

**Figure E-3: Buffer_stress Test Pattern Hardware Implementation Example**

Each slice consists of a constant-color (all components = 0) section that is followed by a noisy section. The number of pixels in the constant-color section is 0 for the first slice and increases by a fixed increment for each slice (note that in this context, a “pixel” represents two pixels from the original image when running in Native 4:2:2 or Native 4:2:0 mode). The noise pattern is an alternating pattern of a variable number of randomly generated pixels, followed by six constant-color pixels in which the constant color has component values of only 0 –or– the maximum possible value (2bpc – 1), as selected by bits from a PRBS generator. The number of consecutive randomly generated pixels is a random number from 1 to 32.

The PRBS generator creates one 16-bit value per pixel, and then advances each pixel in raster order. The PRBS generator ignores slice boundaries. It is straightforward to create a 32- or 64-bit PRBS generator using parallel design techniques, so that multiple pixels/clock can be generated using design parallelism. Generating pixels for multiple slices/line in parallel is also possible by using a different seed value for each slice; however, care must be taken to ensure correct seed initialization for each line.

For YCbCr modes, the pixels are not necessarily within the RGB cube because the component values are random and independent. Some hardware implementations might not have the capability to represent pixels outside of the RGB cube, thus testing using YCbCr stress images might not be possible.

The generate_stress_image.c file included with the CTG package contains a bit-accurate implementation of the test picture. The usage is as follows:

```
generate_stress_image <outfile> <bpc> <pic_width> <pic_height> <slice_width> <slice_height> <mode>
```

where:

- `<outfile>` = Output file name (.ppm for RGB output;
  .yuv for 4:2:2 or 4:2:0 YUV output)
- `<bpc>` = Number of bits per component
- `<pic_width>` = Picture width
- `<pic_height>` = Picture height
- `<slice_width>` = Slice width
- `<slice_height>` = Slice height
- `<mode>` = Type of picture to generate (RGB for an RGB picture, Y444 for a 4:4:4 YCbCr picture, Y422 for a 4:2:2 YCbCr picture, –or– Y420 for a 4:2:0 YCbCr picture)

Figure E-4 is an example image that is output from the generate_stress_image code that would be used to test the Buffer Control Coverage. This example is generated with the following command line with parameters:

```
buffer_stress_image.ppm 10 1920 1080 960 108 RGB
```

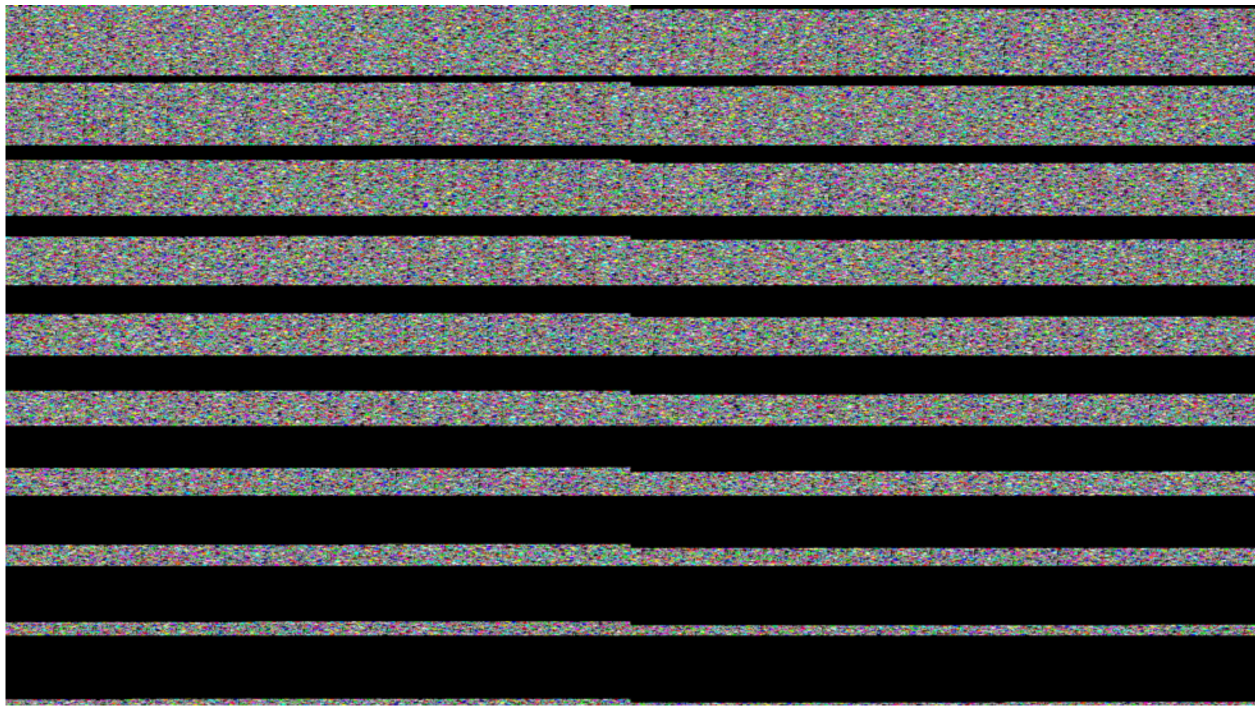

**Figure E-4: Buffer_stress Test Image (RGB, 1920x1080, 960x108 Slices) Example**

# F Using .cfg Files with Test Pictures

The following file is the common .cfg file that can be used for running the different test pictures through the DSC software model when using the automated test parameter generation (`GENERATE_RC_PARAMETERS`) option. See the notes below the sample .cfg file source for information regarding how to set different parameters for each of the tests provided within this Guideline.

*DSC Tools Application Note* code should be used so that the `GENERATE_RC_PARAMETERS` parameter can be used to generate appropriate RC parameters; otherwise, the RC parameters that can be derived from the PPS spreadsheet must be specified in the .cfg file. By including the `SLICE_HEIGHT` parameter as shown below, the automatically generated parameters will use that as a slice height accordingly.

Transports may have their own PPS parameter needs. Such needs should be followed in the .cfg files because any change to the PPS parameters will affect the encode/decode process and therefore the CRCs.

```
// This is a config file for the Display Stream Compression model
SRC_LIST   test_list.txt  // this file holds the name of the picture file(s) to be processed

FUNCTION             0    // 0=encode/decode (no bitstream out), 1=encode only, 2=decode only

SLICE_WIDTH       1920     // Set to the slice width being tested
SLICE_HEIGHT       108    // Set to the slice height being tested

LINE_BUFFER_BPC     16    // Set according to transport requirements
SIMPLE_422           0    // Set to 1 for simple 4:2:2 mode
NATIVE_422           0    // Set to 1 for native 4:2:2 mode
NATIVE_420           0    // Set to 1 for native 4:2:0 mode
USE_YUV_INPUT        0    // Set based on whether YCbCr or RGB is being tested
BLOCK_PRED_ENABLE    1    // To disable block prediction, clear this to 0
VBR_ENABLE           0    // Set to 1 for VBR mode

BITS_PER_PIXEL       8    // Set to desired bpp value
BITS_PER_COMPONENT   8    // Set to desired bpc value
// DPX read options (the following work well for most modes for GM/IM, some anomalies
are autodetected)
DPXR_PAD_ENDS        1    // Pad to 32-bit boundaries
DPXR_DATUM_ORDER     1
DPXR_FORCE_BE        0
SWAP_R_AND_B         1
```

```
// DPX write options (the following work well for most modes for GM/IM)
DPXW_PAD_ENDS        1    // Required to output RGB to XNView 1.99 (but not YUV!)
DPXW_DATUM_ORDER     1
DPXW_FORCE_PACKING   0    // Method to use for 10 & 12-bit data
SWAP_R_AND_B_OUT     1
     // Select the RC settings file based on the BPC & BPP above:
GENERATE_RC_PARAMETERS 1
```

## F.1 SRC_LIST

SRC_LIST points to a file that holds the source picture file name. Be sure to always include the file name extension (i.e., .dpx or .ppm).

## F.2 SLICE_WIDTH and SLICE_HEIGHT

SLICE_WIDTH and SLICE_HEIGHT indicate the slice's pixel width and height, respectively. If width or height are not declared by using // in front of the starting key word, the code will automatically make the width/height match that of the input picture. The sample code shows the SLICE_WIDTH not set so that it equals the picture's actual width. The SLICE_HEIGHT is shown as eight lines.

**Table F-1: Recommended Slice Dimensions to Be Entered into the .cfg Files for Testing[a]**

| Picture | Test | SLICE_WIDTH | SLICE_HEIGHT |
|---|---|---|---|
| DSC1_2-Test_Image_nnbpc.dpx | Core | 1920 | 108 |
| DSC1_2-Test_Image_08bpc - 4 corrupted bits.dsc | Slice Isolation | 1920 | 8 |
| DSC1_2-Buffer_stress | Buffer_stress | See Note a | See Note a |

a. *For designs that support more than 1 slice/line, see Section 4.3.1 for recommendations for additional SLICE_WIDTH runs to test. For DSC1_2-Buffer_stress, SLICE_WIDTH should follow the implementation's capabilities and SLICE_HEIGHT should be chosen so that at least eight slices are present.*

## F.3 BLOCK_PRED_ENABLE

If Block prediction is supported, run the test with BLOCK_PRED_ENABLE set to 1 (maximum coverage). If Block prediction is not supported, only run the test with BLOCK_PRED_ENABLE cleared to 0.

## F.4 VBR_ENABLE

This Guideline does not currently test for VBR mode; therefore, clear VBR_ENABLE to 0. If VBR mode is supported, testing may be conducted with VBR_ENABLE set to 1.

***Note:*** *If this case is enabled, the tester shall determine coverage.*

# G Main Contributor History (Previous Revisions)

**Table G-1: Main Contributor History (Previous Revisions)**

| Company | Name | Contribution | Revision |
|---|---|---|---|
| Advanced Micro Devices, Inc. | Jim Hunkins | Author,<br>Co-chair DSC Task Group | 1.0 |
| Avatar Tech Pubs | Trish McDermott | Technical Writer/Editor | 1.0 |
| Broadcom Inc. | Fred Walls | Technical Content | 1.0 |
| Hardent, Inc. | Simon Bussières | Technical Content | 1.0 |
| NVIDIA Corp. | David Stears | Technical Content | 1.0 |
| Qualcomm, Inc. | James Goel | Author | 1.0 |
|  | Harris Chan | Technical Content | 1.0 |
| Samsung Electronics Co., Ltd. | Dale Stolitzka | Reviewer,<br>Chair DSC Task Group | 1.0 |