|  |  |
| --- | --- |
| Joint Collaborative Team on Video Coding (JCT-VC)  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  26th Meeting: Geneva, CH, 12–20 January 2017 | Document: JCTVC-Z1020-v2 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Common Test Conditions for HDR/WCG video coding experiments** | | |
| *Status:* | Approved Output Document of JCT-VC | | |
| *Purpose:* | Report of agreed test conditions | | |
| *Author(s) or Contact(s):* | E. François, J. Sole, J. Ström, P. Yin | Tel: Email: | [edouard.francois@technicolor.com](mailto:edouard.francois@technicolor.com) [pyin@dolby.com](mailto:pyin@dolby.com) [joels@qti.qualcomm.com](mailto:joels@qti.qualcomm.com) [jacob.strom@ericsson.com](mailto:jacob.strom@ericsson.com) |
| *Source:* | JCT-VC | | |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Abstract

This document defines common test conditions and software reference configurations to be used in the context of experiments for JCT-VC HDR/WCG video coding experiments.

# Introduction

Common test conditions (CTC) are desirable to conduct experiments in a well-defined environment and ease the comparison of the outcome of experiments. This document defines the CTC for the HDR/WCG Video Coding Experiments. There are two anchors defined in HDR CTC: 1) NCL Y′CbCr and 2) ICTCP. Input contributions to JCT-VC should provide a set of results as complete as possible that apply to the proposal compared to the corresponding anchor. Results should be reported using the attached Excel sheet.

The following sections define test sequences, encoder configurations, and the pre- and post-processing options to be used.

# Test material

## File exchange formats

The filenames are specified as follows:

Name\_Resolution\_Fps\_Format\_ContentPrimaries\_ContainerPrimaries\_ChromaFormat\_xxx.yyy

with

* Name: sequence name
* Resolution: picture size (e.g. 1920x1080p)
* Fps: frame rate in frames per second
* Format: format of the samples (e.g. ff for 32-bit floating point, hf for half-float 16-bit floating point, 10 for 10-bit integer)
* Content primaries: colour primaries of the colour volume of the content, e.g. ITU-R Recommendations BT.709 and BT.2100, SMPTE ST 428-1:2006 P3 with D65 white point (P3D65)
* Container primaries: colour primaries of the container (when different from the content primaries), e.g. ITU-R Recommendations BT.709 and BT.2100, SMPTE ST 428-1:2006 P3 with D65 white point (P3D65)
* Chroma format: e.g. 4:2:0, 4:2:2, or 4:4:4 (when applicable; for instance TIFF format involves 4:4:4 interleaved)
* xxx: frame number (when applicable)
* yyy: exr, tif, tiff or yuv

## Test sequences

All the test sequences have the following characteristics:

* Resolution: 1920x1080 progressive
* Colour format: RGB 4:4:4
* Container: BT.2020 or P3D65 depending on the content

The test sequences considered for the evaluation tests are listed in Table 1.

Table . HDR test sequences.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Class** | **seq** | **gamut** | **TF** | **Sequence name** | **fps** | **Frames** |
| A | S00 | BT.709 | Linear | FireEater2Clip4000r1\_1920x1080p\_25\_hf\_709\_ct2020\_444\_xxx.exr | 25 | 0-199 |
|  | S02 | BT.709 | Linear | Market3Clip4000r2\_1920x1080p\_50\_hf\_709\_ct2020\_444\_xxx.exr | 50 | 0-399 |
| B | S04 | P3D65 | PQ12b | BikeSparklersClip4000\_1920x1080p\_24\_12\_P3\_ct2020\_xxx.tif 1 | 24 | 23-128 +  216-340 |
| C | S05 | P3D65 | PQ12b | ShowGirl2TeaserClip4000\_1920x1080p\_25\_12\_P3\_ct2020\_xxx.tif | 25 | 0-338 |
| D | S06 | P3D65 | PQ12b | StEM\_MagicHour\_1920x1080p\_24\_12\_P3\_xxx.tif 2 | 24 | 3527 to 3608 + 3609 to 3803 + 3804 to 3887 |
|  | S07 | P3D65 | PQ12b | StEM\_WarmNight\_1920x1080p\_24\_12\_P3\_xxx.tif 2 | 24 | 6280 to 6481 + 6482 to 6640 |
| G | S08 | BT.709 | Linear | BalloonFestival\_1920x1080p\_24\_hf\_709\_ct2020\_444\_xxx.exr | 24 | 0-239 |
| H | S10 | BT.709 | Linear | EBU\_04\_Hurdles\_1920x1080p\_25\_hf\_709\_ct2020\_444\_xxx.exr | 100 | 0-499 |
|  | S11 | BT.709 | Linear | EBU\_06\_Starting\_1920x1080p\_25\_hf\_709\_ct2020\_444\_xxx.exr | 100 | 0-499 |
| A | S12 | BT.709 | Linear | Sunrise\_1920x1080p\_25\_hf\_709\_ct2020\_444\_xxx.exr | 25 | 0-199 |
| B | S13 | P3D65 | PQ12b | GarageExit\_1920x1080p\_24\_12\_P3\_ct2020\_xxx.tif | 24 | 0-287 |

1 for BikeSparklers, only 1st and 3rd cuts are kept ([023-128], [216-340])

2 for StEM sequences, the successive cuts of S06 and S07 are considered

Material originally in a P3D65 container will be evaluated after direct conversion into a BT.2100 PQ container, as specified in section 4.2.2.2 for Y′CbCr and in section 5.2.2.2 for ICTCP.

Notes:

* StEM (S06, S07) and ShowGirl2Teaser (S05) sequences use letter box. Cropping is required for derivation of objective metrics.
  + For StEM, the objective metrics must be computed in cropping window delimited by pixels (column=0, row=140) to (column=1919, row=939).
  + For ShowGirl2Teaser, the objective metrics must be computed in cropping window delimited by pixels (column=10, row=10) to (column=1909, row=1069).

Metrics configuration files HDRMetric\_CfE\_ShowGirl.cfg and HDRMetric\_CfE\_StEM.cfg are provided in the “bin/JCTVC\_CTC\_cfgFiles” folder of the HDRTools reference software package for Y′CbCr and ICTCP.

* TIFF sequences must be converted, before encoding, to Y′CbCr 4:2:0 10b format, as described in section 4.2.2; or ICTCP 4:2:0 10b format, as described in section 5.2.2.

*Comments:*

***Sequences with cuts/fading***

* *BikeSparklers: only 1st and 3rd cuts are kept ([023-128] + [216-340])*
* *Sequences with cuts are coded with constant qp*
  + *Anchors bitrate has to be matched by proposals*
  + *Objective metrics are computed for each cut*
  + *Viewing done on the complete re-concatenated sequences*

## Access to test sequences

Sequences from classes A, G and H are accessible at the following location using login/password credentials sent to JCT-VC participants (qualified members of the JCT-VC may contact [Gary Sullivan](mailto:garysull@microsoft.com) or [Jens-Rainer Ohm](mailto:ohm@ient.rwth-aachen.de) to obtain access):

* <ftp://ftp.ient.rwth-aachen.de/testsequences/testset_hdr>

Note that test sequences from the EBU (i.e. with the string ‘EBU’ included in the test sequence name) are password protected. The password can be obtained by registering at the following location:

* <https://tech.ebu.ch/EBU_SVT_Public_Test_Sequences>

Access information to sequence from classes B can be obtained from Walt Husak ([WJH@dolby.com](mailto:WJH@dolby.com)).

Access information to sequences from classes C can be obtained from Jan Fröhlich ([jan@fr.oehli.ch](mailto:jan@fr.oehli.ch)).

Sequences from classes D are clips extracted from content in its complete version. Details to access this content can be found at <http://www.dcimovies.com/2014_StEM_Access/>. Special care has to be made about the copyright license. The proponents are invited to perform on their own clips extraction from the complete content.

# Anchors coding conditions

Test conditions not explicitly noted in this document are aligned with the HM Common Test Conditions.

For sequences S10 and S11, even if the frame rate of the source content is 100, the intra refresh period must be set to 48.

Y′CbCr anchors are generated per the process described in section 4.1.2. The anchor bitstreams are generated based on the HEVC Main 10 Profile, using BT.2100 colour primaries and NCL Y′CbCr colour space conversion, the PQ Transfer Function, fixed QP values.

ICTCP anchors are generated per the process described in section 5.1.2. The anchor bitstreams are generated based on the HEVC Main 10 Profile, using BT.2100 colour primaries and ICTCP colour space conversion, the PQ Transfer Function, fixed QP values.

Figure 1 shows the end-to-end coding and decoding chain used in generating HDR PQ Y′CbCr 4:2:0 anchors and evaluating visual quality. Figure 2 shows the end-to-end coding and decoding chain used in generating HDR PQ ICTCP 4:2:0 anchors and evaluating visual quality.

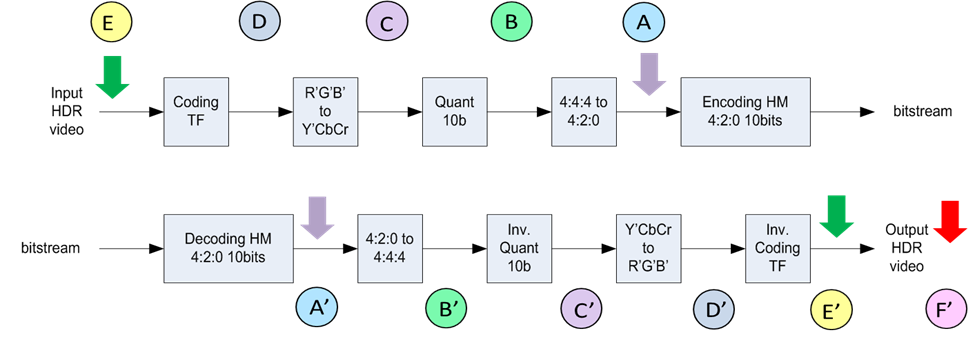


Figure . End to end coding and decoding chain for Y′CbCr 4:2:0 video.



A

B

E

F’

A’

B’

E’

C

C’

D

D’

Input

HDR

video

Coding

TF

L’M’S’

to

ICTCP

Quant

10

b

4

:

4

:

4

to

4

:

2

:

0

Encoding HM

4

:

2

:

0 10

bits

bitstream

bitstream

Decoding HM

4

:

2

:

0 10

bits

4

:

2

:

0

to

4

:

4

:

4

Inv

.

Quant

10

b

ICTCP

to

L’M’S’

Inv

.

Coding

TF

Output

HDR

video

RGB

to

LMS

LMS

to

RGB

Figure . End to end coding and decoding chain for ICTCP 4:2:0 video.

Objective visual quality evaluations

Comparisons will be made using objective measurement(s) from the decoded video provided by all proponents. The objective measurements will be made between the points EE′ at various bit rates. An excel sheet is provided in attachment to report the objective metrics results.

For sequences with cuts, as mentioned in section 3, the encoding is performed on each cut (sub-sequence). The viewing is done with the re-concatenated of the sub-sequences.

The template for reporting the performance results is provided in the attached file   
JCTVC-CTCs\_HDR\_template.xls.

Subjective visual quality evaluation

Formal viewing tests (at point F′ in Figure 1 and Figure 2) will be performed when necessary.

# Anchor generation process for Y′CbCr 420 video

## Software versions

### Conversion software HDRConvert

The different conversions required in the generation of anchors are done using the HDRConvert tool. This tool is accessible to qualified members of the JCT-VC at the following location using login/password credentials sent to JCT-VC participants (qualified members of the JCT-VC may contact [Gary Sullivan](mailto:garysull@microsoft.com) or [Jens-Rainer Ohm](mailto:ohm@ient.rwth-aachen.de) to obtain access):

<https://gitlab.com/standards/HDRTools/>

HDRTools-v0.15 software or newer is expected to be used for experiments.

### HDR anchor coding process

The HDR anchors are based on the HEVC Main 10 profile, which implies encodings using 4:2:0 subsampling and 10 bits per channel.

The encoding is performed with the HM recommended in the JCT-VC Common Test Conditions. HM-16.15 software or newer is expected to be used for experiments.

The HM must be configured in 4:2:0 with the following settings:

* The macro RExt\_\_HIGH\_BIT\_DEPTH\_SUPPORT is set to 1, which results in
  + FULL\_NBIT set to 1
  + RExt\_\_HIGH\_PRECISION\_FORWARD\_TRANSFORM set to 1
* Using Random Access (RA) configuration from HEVC common test conditions.

The following example configuration files are in the “cfg/cfg\_HDR” folder of the HM reference software package:

* encoder\_randomaccess\_main10\_HDR\_YCbCr\_classAGH.cfg and
* encoder\_randomaccess\_main10\_HDR\_YCbCr\_classBCD.cfg

## Conversion and inverse conversion processes

### RGB linear-light format with BT.709 or BT.2100 primaries for HDR anchor

If the input is in a half float 4:4:4 RGB linear-light format (e.g. OpenEXR), the bit streams shall be generated using the coding / decoding chain illustrated in Figure 3.

The conversion to 4:2:0 10 bits Y′CbCr is obtained with the HDRConvert tool using the example configuration file HDRConvertEXR2020ToYCbCr420.cfg (for the BT.2100 content container primaries) provided in the “bin/JCTVC\_CTC\_cfgFiles” folder of the HDRTools reference software package. The parameter ClosedLoopConversion must be set to 8.

This conversion consists of the following steps:

* Convert half precision floating point data to single precision floating point data (not illustrated).
* Map using the PQ transfer function (PQ-TF) from RGB (float) to R′G′B′ (float) by invoking Section 4.4.1.
* Convert from R′G′B′ (float) to Y′CbCr by invoking section 4.4.2 if the sequence is in BT.709 container, or section 4.4.3 if the sequence is in BT.2100 container.
* Quantize from Y′CbCr (float) into DY′DCbDCr (10bit) by invoking Section 4.5, with BitDepthY and BitDepthC set to 10.
* Downsample both chroma components from 4:4:4 DY′DCbDCr (10bit) to 4:2:0 DY′DCbDCr (10bit) by invoking Section 4.6.
* Perform luma adjustment using RGB and the subsampled DCbDCr signals as inputs to get the DY′ signal resulting in 4:2:0 DY′DCbDCr (10bit) by invoking Section 4.10.

Note: Dxx stands for Digitized version of signal xx. R′G′B′ or Y′CbCr stands for normalized value within [0,1] (R′G′B′ Y′) and [-0.5, 0.5] (CbCr).

The reverse conversion is obtained with the HDRConvert tool using the configuration file HDRConvertYCbCr420ToEXR2020.cfg (for the BT.2100 content container primaries) provided in the “bin/JCTVC\_CTC\_cfgFiles” folder of the HDRTools reference software package.

The process consists of the following steps:

* Upsample both chroma components from 4:2:0 DY′DCbDCr (10bit) to 4:4:4 DY′DCbDCr (10bit) by invoking Section 4.7.
* Inverse quantize from DY′DCbDCr (10bit) into Y′CbCr (float) by invoking section 4.8, with BitDepthY and BitDepthC set to 10.
* Convert from Y′CbCr (float) to R′G′B′ (float) by invoking section 4.9.1 if the sequence is in a BT.709 container, or section 4.9.2 if the sequence is in a BT.2100 container.
* Inverse map using the inverse PQ-TF from R′G′B′ (float) to RGB (float) by invoking Section 4.9.3.
* Convert, if needed, the data from single precision floating point numbers to half precision floating point numbers using appropriate rounding operations (not illustrated).



Figure . Simplified encoding / decoding chains when input HDR video is RGB linear light.

### Input 4:4:4 DR′DG′DB′ PQ-TF 12bit for HDR Anchor

If the input is using a 4:4:4 PQ-TF 12bit DR′DG′DB′ format, the bit streams shall be generated using the coding / decoding chain illustrated in Figure 4.

#### Input 4:4:4 DR′DG′DB′ PQ-TF 12bit in BT.2100 container

The conversion to 4:2:0 10 bits Y′CbCr is obtained with the HDRConvert tool using the configuration file HDRConvertBT2020TiffToYCbCr420.cfg provided in the “bin/JCTVC\_CTC\_cfgFiles” folder of the HDRTools reference software package. The parameter ClosedLoopConversion must be set to 8.

The process consists of the following steps:

* Convert from 12bit DR′DG′DB′ into normalized R′G′B′ by invoking section 4.12.1.
* Convert from R′G′B′ (BT.2100) to linear RGB by applying the inverse transfer function (inversePQ-TF) as described in Section 4.9.3. The linear RGB will be used in the last step (luma adjustment).
* Convert from R′G′B′ (BT.2100) to Y′CbCr by invoking section 4.4.3.
* Quantize from Y′CbCr (float) into DY′DCbDCr (10bit) by invoking Section 4.5, with BitDepthY and BitDepthC set to 10.
* Downsample both chroma components from 4:4:4 DY′DCbDCr (10bit) to 4:2:0 DY′DCbDCr (10bit) by invoking Section 4.6.
* Perform luma adjustment using the linear RGB obtained in the second step and the subsampled DCbDCr signals obtained in the previous step, as described in Section 4.10. The output is a DY′ signal resulting in 4:2:0 DY′DCbDCr (10bit).

The reverse conversion is obtained by applying the following steps and the output is DR′DG′DB′ PQ-TF 12b in BT.2100 container. It can be obtained with the HDRConvert tool using the configuration file HDRConvertYCbCr420ToBT2020Tiff.cfg provided in the “bin/JCTVC\_CTC\_cfgFiles” folder of the HDRTools reference software package.

* Upsample both chroma components from 4:2:0 DY′DCbDCr (10bit) to 4:4:4 DY′DCbDCr (10bit) by invoking Section 4.7.
* Inverse quantize from DY′DCbDCr (10bit) into Y′CbCr (float) by invoking section 4.8, with BitDepthY and BitDepthC set to 10.
* Convert from Y′CbCr (float) to R′G′B′ (float) (BT.2100) by invoking section 4.9.2.
* Quantize normalized R′G′B′ into 12bit DR′DG′DB′ in BT.2100 container by invoking section 4.12.2.



Figure 4. Simplified encoding / decoding chains when the input HDR video is in 4:4:4 R′G′B′ PQ-TF 12bit.

#### Input 4:4:4 DR′DG′DB′ PQ-TF 12bit in P3D65 container

The conversion to 4:2:0 10 bits Y′CbCr is obtained with the HDRConvert tool using the configuration file HDRConvertP3D65TiffToYCbCr420.cfg provided in the “bin/JCTVC\_CTC\_cfgFiles” folder of the HDRTools reference software package. The parameter ClosedLoopConversion must be set to 8.

The process consists of the following steps:

* Convert from PQ-TF P3D65 12bit DR′DG′DB′ into normalized R′G′B′ by invoking section 4.12.1.
* Convert R′G′B′ PQ-TF P3D65 to RGB P3D65 by invoking section 4.9.3.
* Convert from RGB P3D65 to RGB BT.2100 by invoking section 4.11.4.
* Convert from RGB BT.2100 to R′G′B′ PQ-TF BT.2100 by invoking section 4.4.1.
* Convert from R′G′B′ (BT.2100) to Y′CbCr by invoking section 4.4.3.
* Quantize from Y′CbCr (float) into DY′DCbDCr (10bit) by invoking Section 4.5, with BitDepthY and BitDepthC set to 10.
* Downsample both chroma components from 4:4:4 DY′DCbDCr (10bit) to 4:2:0 DY′DCbDCr (10bit) by invoking Section 4.6.
* Perform luma adjustment using the linear RGB P3D65 obtained in the second step and the subsampled DCbDCr signals obtained in the previous step, as described in Section 4.10. The output is a DY′ signal resulting in 4:2:0 DY′DCbDCr (10bit).

## Colour space conversion from RGB to XYZ

### RGB with BT.709 primaries to XYZ

* + X = 0.412391 \* R + 0.357584 \* G + 0.180481 \* B
  + Y = 0.212639 \* R + 0.715169 \* G + 0.072192 \* B
  + Z = 0.019331 \* R + 0.119195 \* G + 0.950532 \* B

### RGB with BT.2100 primaries to XYZ

* + X = 0.636958 \* R + 0.144617 \* G + 0.168881 \* B
  + Y = 0.262700 \* R + 0.677998 \* G + 0.059302 \* B
  + Z = 0.000000 \* R + 0.028073 \* G + 1.060985 \* B

### RGB with P3D65 primaries to XYZ

* + X = 0.486571 \* R + 0.265668 \* G + 0.198217 \* B
  + Y = 0.228975 \* R + 0.691739 \* G + 0.079287 \* B
  + Z = 0.000000 \* R + 0.045113 \* G + 1.043944 \* B

## Colour transformation from RGB to Y′CbCr

### Conversion from RGB to R′G′B′

* + R′ = PQ\_TF(max(0, min(R/10000,1)) )
  + G′ = PQ\_TF(max(0, min(G/10000,1)) )
  + B′ = PQ\_TF(max(0, min(B/10000,1)) )

with

### R′G′B′ with BT.709 primaries to Y′CbCr

The ITU-R BT.709 standard specifies the following conversion process from R′G′B′ to Y′CbCr (non-constant luminance representation):

* + Y′ = 0.2126 \* R′ + 0.7152 \* G′ + 0.0722 \* B′

The above can also be implemented using the following approximate conversion that avoids the division for the Cb and Cr components:

* + Y′ = 0.212600 \* R′ + 0.715200 \* G′ + 0.072200 \* B′
  + Cb = −0.114572 \* R′ − 0.385428 \* G′ + 0.500000 \* B′
  + Cr = 0.500000 \* R′ − 0.454153 \* G′ − 0.045847 \* B′

The above method, without division, is the method recommended to use for participation in this activity and for the generation of the current test anchors.

### R′G′B′ with BT.2100 primaries to Y′CbCr

The ITU-R BT.2100 standard specifies the following conversion process from R′G′B′ to Y′CbCr (non-constant luminance representation):

* + Y′ = 0.2627 \* R′ + 0.6780 \* G′ + 0.0593 \* B′

The above can also be implemented using the following approximate conversion that avoids the division for the Cb and Cr components:

* + Y′ = 0.262700 \* R′ + 0.678000 \* G′ + 0.059300 \* B′
  + Cb = −0.139630 \* R′ − 0.360370 \* G′ + 0.500000 \* B′
  + Cr = 0.500000 \* R′ − 0.459786 \* G′ − 0.040214 \* B′

The above method, without division, is the method recommended to use for participation in this activity and for the generation of the current test anchors.

## Quantization from Y′CbCr into DY′DCbDCr

This process quantizes the input Y′CbCr signal into a signal of bit-depth BitDepthY for the Y component and BitDepthC for the chroma components (Cb, Cr).

with

Round( x ) = Sign( x ) \* Floor( Abs( x ) + 0.5 )

Sign ( x ) = -1 if x < 0, 0 if x=0, 1 if x > 0

Floor( x ) the largest integer less than or equal to x

Abs( x ) = x if x>=0, -x if x<0

Clip1Y( x ) = Clip3( 0, ( 1 << BitDepthY ) − 1, x )

Clip1C( x ) = Clip3( 0, ( 1 << BitDepthC ) − 1, x )

Clip3( x,y,z ) = x if z<x, y if z>y, z otherwise

## Chroma downsampling from 4:4:4 to 4:2:0

The chroma samples alignment is as follows:



|  |  |  |
| --- | --- | --- |
| Phase k | Coefs c1[k]  4:4:4 🡪 4:2:2  (phase=0) | Coefs c2[k]  4:2:2 🡪 4:2:0  (phase=0) |
| −1 | 1 | 1 |
| 0 | 6 | 6 |
| 1 | 1 | 1 |

* Define shift = 6 and offset = 32.
* Let *H* and *W* be the input picture height and width in chroma samples. For *i*= 0..*H*−1, *j*= 0..*W*/2−1, the intermediate samples *f*[ *i*][ *j*] are derived from the input samples *s*[ *i*][ *j*] as follows:

with Clip3*( x,y,z ) = x* if *z<x,* y if *z>y, z* otherwise

* For *i*= 0..*H*/2-1, *j* = 0..*W*/2-1, the output samples *r*[ *i*][ *j*] are derived from the intermediate samples *f*[ *i*][ *j*] as follows:

## Chroma upsampling from 4:2:0 to 4:4:4 (Y′CbCr domain)

The upsampling filter used is the same for both horizontal and vertical processes. First, vertical filtering is applied on the 4:2:0 picture, then horizontal filtering.

Filter coefficients values are as follows

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Phase | −2 | −1 | 0 | 1 |
| Coef c[k] | −4 | 36 | 36 | −4 |

Define shift1 = 6, offset1 = 32, shift2 = 12, offset2 = 2048.

Let H and W be the input picture height and width in chroma samples. For i = 0..H-1, j = 0..W-1, the intermediate samples f[ i ][ j ] are derived from the input samples s[ i ][ j ] as follows:

For i = 0..2\*H-1, j = 0..W-1, the output samples r[ i ][ j ] are derived from the intermediate samples f[ i ][ j ] as follows:

## Inverse Quantization from DY′ DCbDCr into Y′CbCr

This process dequantizes the input signal represented on BitDepthY bits for the Y component and BitDepthC bits for the chroma components (Cb, Cr) into a (float) signal Y′CbCr.

with

ClipY′ (x) = Clip3 ( 0, 1.0, x)

ClipC (x) = Clip3 ( -0.5, 0.5, x)

Clip3( x,y,z ) = x if z<x, y if z>y, z otherwise

## Colour transformation from Y′CbCr to RGB

### Y′CbCr to R′G′B′ with BT.709 primaries

* + R′ = clipRGB(Y′ + 1.57480 \* Cr)
  + G′ = clipRGB(Y′ – 0.18733 \* Cb – 0.46813 \* Cr)
  + B′ = clipRGB(Y′ + 1.85563 \* Cb)

### Y′CbCr to R′G′B′ with BT.2100 primaries

* + R′ = clipRGB(Y′ + 1.47460 \* Cr)
  + G′ = clipRGB(Y′ – 0.16455 \* Cb – 0.57135 \* Cr)
  + B′ = clipRGB(Y′ + 1.88140 \* Cb)

with clipRGB( x ) = Clip3( 0, 1, x )

Clip3( x,y,z ) = x if z<x, y if z>y, z otherwise

### Conversion from R′G′B′ to RGB

* + R = 10000\*inversePQ\_TF(R′)
  + G = 10000\*inversePQ\_TF(G′)
  + B = 10000\*inversePQ\_TF(B′)

with

## Luma Adjustment

This section describes the process of calculating the luma code word DY′. The Y′ obtained in Section 4.4 (and hence the quantized value obtained from it) can generate pixels for which the luminance deviates from the original pixel, giving rise to strong artifacts. The luma adjustment process selects a luma code word DY′ that makes the decoded pixel have a more correct luminance. Luminance is defined as the Y-component of the CIE1931 XYZ colour that is obtained by converting the original linear RGB value as described in section 4.3.

First the desired, or original, luminance is calculated from the linear RGB values using Section 4.3:

where , and depend on the input container. For instance, if the input container is Rec. BT.709, the values are , and in accordance with the middle equation of Section 4.3.1.

Second, the upsampled versions of Cb and Cr are obtained for each pixel. This is done by first upsampling the subsampled DCbDCr signals according to Section 4.7 followed by inverse quantization according to Section 4.8. Note that since these values have been downsampled and then upsampled, they differ from the ones obtained in Section 4.4.1, and hence hats are used to denote them according to and .

Third, interval halving is used to find the value that results in a luminance value that best matches the original luminance value . To that end, a starting interval [64, 940] is associated with each pixel[[1]](#footnote-1). A candidate code word in the middle of this interval is then tried: = 502. A floating point candidate is obtained by using the first equation of Section 4.8:

This is now used to calculate the R′, G′ and B′ values by applying Section 4.9.2, which is repeated here for the convenience of the reader:

* + = clipRGB( + 1.47460 \* )
  + = clipRGB( – 0.16455 \* – 0.57135 \* )
  + = clipRGB( + 1.88140 \* )

These values are then converted to linear RGB by invoking Section 4.9.3, and then the candidate luminance value is calculated as

Now, if the candidate luminance value is too high, i.e., , the interval is updated to the lower half of the starting interval, i.e., [64, 502]. If on the other hand , the upper half [502, 940] is used. A new candidate is now computed as the middle of the updated value, and the process continues until, after at most 10 steps, we end up with an interval [a, b] of size one, ie., b = a+1.

Finally, a selection between a and b is done by computing and according to the process above. is selected if

and is selected otherwise, where is defined in Section 4.4.1

## Colour space container conversion

This section describes the process of converting RGB samples encoded in and limited by one particular colour space, e.g. ITU-R Recommendation BT.709, BT.2100, or P3D65, into another RGB colour space specified with different primaries. We are particularly interested in the conversion of RGB BT.709 as well as RGB P3D65 samples into RGB BT.2100 samples and vice versa.

### RGB conversion: from BT.2100 to BT.709

It is essential when displaying data on a display that can only operate with BT.709 primaries to appropriately convert them in that space before display. To do so, this can be done using either a two-step conversion process, from the current colour space of the data (i.e. ITU-R BT.2100) to XYZ, followed by a subsequent conversion to BT.709, or using a single step process. In particular, the two-step conversion process can be applied as follows:

* Conversion from R2020G2020B2020 to XYZ:
  + X = 0.636958 \* R2020 + 0.144617 \* G2020 + 0.168881 \* B2020
  + Y = 0.262700 \* R2020 + 0.677998 \* G2020 + 0.059302 \* B2020
  + Z = 0.000000 \* R2020 + 0.028073 \* G2020 + 1.060985 \* B2020
* Conversion from XYZ to R709G709B709 (BT.709):
  + R709 = clipRGB( 3.240970 \* X - 1.537383 \* Y - 0.498611 \* Z )
  + G709 = clipRGB( -0.969244 \* X + 1.875968 \* Y + 0.041555 \* Z )
  + B709 = clipRGB( 0.055630 \* X - 0.203977 \* Y + 1.056972 \* Z )

The above could be converted into a single step by combining the two matrix conversions above into a single matrix. This would result in the following, high precision matrix conversion:

* + R709 = clipRGB( 1.660490254890140 \* R2020

− 0.587638564717282 \* G2020 - 0.072851975229213 \* B2020 )

* + G709 = clipRGB( −0.124550248621850 \* R2020

+ 1.132898753013895 \* G2020 − 0.008347895599309 \* B2020 )

* + B709 = clipRGB( −0.018151059958635 \* R2020

− 0.100578696221493 \* G2020 + 1.118729865913540 \* B2020 )

We currently would recommend the single step approach, as above, for the conversion of RGB BT.2100 material to RGB BT.709.

### RGB conversion: from P3D65 to BT.709

This conversion can be done using either a two-step conversion process, from the current colour space of the data (i.e. P3D65) to XYZ, followed by a subsequent conversion to BT.709, or using a single step process. In particular, the two-step conversion process can be applied as follows:

* Conversion from RP3GP3BP3 (P3D65) to XYZ:
  + X = 0.486571 \* RP3 + 0.265668 \* GP3 + 0.198217 \* BP3
  + Y = 0.228975 \* RP3 + 0.691739 \* GP3 + 0.079287 \* BP3
  + Z = 0.000000 \* RP3 + 0.045113 \* GP3 + 1.043944 \* BP3
* Conversion from XYZ to R709G709B709 (BT.709):
  + R709 = clipRGB( 3.240970 \* X − 1.537383 \* Y − 0.498611 \* Z )
  + G709 = clipRGB( −0.969244 \* X + 1.875968 \* Y + 0.041555 \* Z )
  + B709 = clipRGB( 0.055630 \* X − 0.203977 \* Y + 1.056972 \* Z )

The above could be converted into a single step by combining the two matrix conversions above into a single matrix. This would result in the following, high precision matrix conversion:

* + R709 = clipRGB( 1.224939741445000 \* RP3

- 0.224939599120000 \* GP3 − 0.000001097215000 \* BP3 )

* + G709 = clipRGB( −0.042056249524000 \* RP3

+ 1.042057784075000 \* GP3 − 0.000000329788000 \* BP3 )

* + B709 = clipRGB( −0.019637688845000 \* RP3

− 0.078636557327000 \* GP3 + 1.098273664879000 \* BP3 )

We currently would recommend the single step approach, as above, for the conversion of RGB P3D65 material to RGB BT.709.

### RGB conversion: from BT.709 to BT.2100

The process to convert RGB BT.709 samples to RGB BT.2100 samples is very similar to the process performed earlier for the inverse conversion. A two-step conversion involves first applying the RGB BT.709 to XYZ conversion process, followed by a conversion from XYZ to RGB BT.2100 using the appropriate conversion matrices. In particular, this is done as follows:

* Conversion from R709G709B709 (BT.709) to XYZ
  + X = 0.412391 \* R709 + 0.357584 \* G709 + 0.180481 \* B709
  + Y = 0.212639 \* R709 + 0.715169 \* G709 + 0.072192 \* B709
  + Z = 0.019331 \* R709 + 0.119195 \* G709 + 0.950532 \* B709
* Conversion from XYZ to R2020G2020B2020 (BT.2100)
  + R2020 = clipRGB( 1.716651 \* X – 0.355671 \* Y - 0.253366 \* Z )
  + G2020 = clipRGB( -0.666684 \* X + 1.616481 \* Y + 0.015768 \* Z )
  + B2020 = clipRGB( 0.017640 \* X - 0.042771 \* Y + 0.942103 \* Z )

Similarly, the single step and recommended method is as follows:

* + R2020 = clipRGB( 0.627404078626 \* R709 + 0.329282097415 \* G709 + 0.043313797587 \* B709 )
  + G2020 = clipRGB( 0.069097233123 \* R709 + 0.919541035593 \* G709 + 0.011361189924 \* B709 )
  + B2020 = clipRGB( 0.016391587664 \* R709 + 0.088013255546 \* G709 + 0.895595009604 \* B709 )

### RGB conversion: from P3D65 to BT.2100

The conversion process to convert RGB P3D65 samples to RGB BT.2100 is very similar to the one described in the previous section. Again, this could be done using either a two-step or a single‑step method. The two-step method is as follows:

* Conversion from RP3GP3BP3 (P3D65) to XYZ:
  + X = 0.486571 \* RP3 + 0.265668 \* GP3 + 0.198217 \* BP3
  + Y = 0.228975 \* RP3 + 0.691739 \* GP3 + 0.079287 \* BP3
  + Z = 0.000000 \* RP3 + 0.045113 \* GP3 + 1.043944 \* BP3
* Conversion from XYZ to R2020G2020B2020 (BT.2100):
  + R2020 = clipRGB( 1.716651 \* X − 0.355671 \* Y − 0.253366 \* Z )
  + G2020 = clipRGB( −0.666684 \* X + 1.616481 \* Y + 0.015768 \* Z )
  + B2020 = clipRGB( 0.017640 \* X − 0.042771 \* Y + 0.942103 \* Z )

Similarly, the single step and recommended method is as follows:

* + R2020 = clipRGB( 0.753832826496 \* RP3 + 0.198597635641 \* GP3 + 0.047569409186 \* BP3 )
  + G2020 = clipRGB( 0.045744636411 \* RP3 + 0.941777687331 \* GP3 + 0.012478735611 \* BP3 )
  + B2020 = clipRGB( -0.001210377285 \* RP3 + 0.017601107390 \* GP3 + 0.983608137835 \* BP3 )

## Conversion of HDR TIFF Input files

HDR files provided in TIFF format use a 12 bit non-linearly quantized (using PQ) RGB signal representation. The data are provided using the SDI data range (code values from 16 up to 4076) and may use either the BT.2100 or P3D65 RGB colour space. Converting back and forth from non-linear quantized data to non-linear normalized data is specified in the following sections:

Note: The 12 bit R′G′B′ data is contained in 16 bit tiff container and the 12 most significant bits are used.

### Inverse Quantization from 12bit DR′DG′DB′ into normalized PQ R′G′B′

The inverse quantization from 12bit DR′DG′DB′ into normalized R′G′B′ for PQ is achieved as follows:

* + C′ = (DC′ – range\_low) / (range\_high - range\_low) where C′ = R′, G′, B′

with range\_low = 16 and range\_high = 4076.

### Quantization from normalized R′G′B′ into 12bit PQ DR′DG′DB′

The quantization from normalized R′G′B′ to 12bit DR′DG′DB′ for PQ is achieved as follows:

* + DC′ = Round((C′ + range\_low) \* (range\_high - range\_low)) where C′ = R′, G′, B′

with range\_low = 16 and range\_high = 4076

## HM encoding

The HM encoding is performed with the following features activated using proper configuration settings (as explained below):

* A specific chroma qp offset process is applied to better control the chroma shifting
* A specific control of the luma delta QP is applied to better balance the bitrate between dark and bright areas.

These two changes are described in the following sub-sections.

### Chroma QP offset

In order to get a good balance between bits spent on luma and chroma it may be needed to adjust the QP offsets for Cb and Cr, depending upon the bit rate, and the native colour space. A model is used that assigns Cb and Cr QP offsets based on the luma QP and a factor based on the capture colour space and the representation colour space. The model is expressed as

QPoffset\_Cb = max(−12, round(c\_cb\*(k\*QP+l)))

QPoffset\_Cr = max(−12, round(c\_cr\*(k\*QP+l))),

where c\_cb = 1 if capture colour space is same as representation colour space, c\_cb=1.04 if capture colour space is P3D65 and representation colour space is BT.2100, c\_cb=1.14 if capture colour space is BT.709 and representation space is BT.2100.

Furthermore, c\_cr = 1 if capture colour space is same as representation colour space, c\_cr=1.39 if capture colour space is P3D65 and representation colour space is BT.2100, c\_cr=1.78 if capture colour space is BT.709 and representation space is BT.2100.

Finally, k = −9.46 and l = 9.26.

#### Implementation in HM encoder

HM includes several new configuration parameters to cover k, l, c\_cb and c\_cr in the above equations, see Table 2. Based on the luma QP, a Cb QP offset and a Cr QP offset are determined and used to set the PPS Cb QP and Cr QP offsets.

These parameters are used to calculate the two chroma QP offsets which are then signalled in the PPS. HM also signals multiple PPS units within each coded sequence. This feature is used to signal new chroma QP offsets when the base luma QP changes in order to match bit rate in a fixed QP setting. When a QP change according to the floating point QP occurs for a slice a new PPS is encoded with chroma QP offsets set according to the new base luma QP and referred by the slice. A control parameter WCGPPSEnable is used to control the on/off of these settings and shall be set to 1.

Table . Configuration parameters for HM encoder.

|  |  |
| --- | --- |
| WCGPPSChromaQpScale | This floating point value corresponds to the k parameter in the model |
| WCGPPSChromaQpOffset | This floating point value corresponds to the l parameter in the model |
| WCGPPSCbQpScale | This floating point value corresponds to the c\_cb parameter |
| WCGPPSCrQpScale | This floating point value corresponds to the c\_cr parameter |

To remain compatible with current behavior the two existing parameters (CbQpOffset and CrQpOffset) for setting chroma QP offsets in HM are kept. They are added afterwards to the calculated offsets when the scaling has been done and allows for setting a stronger or weaker chroma qp offset than that specified by the model.

### Average Luma Controlled Adaptive dQP

The combination of the ST 2084 transfer function and the BT.2100 colour space allocates relatively more bits to the darker areas than what is the case for SDR compression (BT.709 transfer function / BT.2100 colour space). In order to get a better balance between dark and bright areas in the image, the QP value is changed according to the average luma value in each 64x64 CTU pixel block.

#### Implementation

For every 64x64 CTU, the average luma value, Laverage, is calculated. An integer version intL of this is then calculated using intL = floor(Laverage + 0.5). The QP to use for a particular 64x64 CTU block is then obtained by adding the picture QP with the dQP obtained by using the look-up table shown in Table 3. The source code in HM is marked with the SHARP\_LUMA\_DELTA\_QP defines. This luma adaptive dQP method is enabled by the following parameter in config file.

LumaLevelToDeltaQPMode: 1 # Change luma delta QP based on average luma

Table . Look-up table of the dQP value from the average of the luma value. As an example, an average luma value of 503 in a 64x64 block would result in a dQP value of −1.

|  |  |
| --- | --- |
| **luma intL range** | **dQP** |
| intL < 301 | 3 |
| 301 ≤ intL < 367 | 2 |
| 367 ≤ intL < 434 | 1 |
| 434 ≤ intL < 501 | 0 |
| 501 ≤ intL < 567 | −1 |
| 567 ≤ intL < 634 | −2 |
| 634 <= intL < 701 | −3 |
| 701 <= intL < 767 | −4 |
| 767 <= intL < 834 | −5 |
| intL >= 834 | −6 |

The data can also be drawn as a diagram, which is shown in Figure 5.

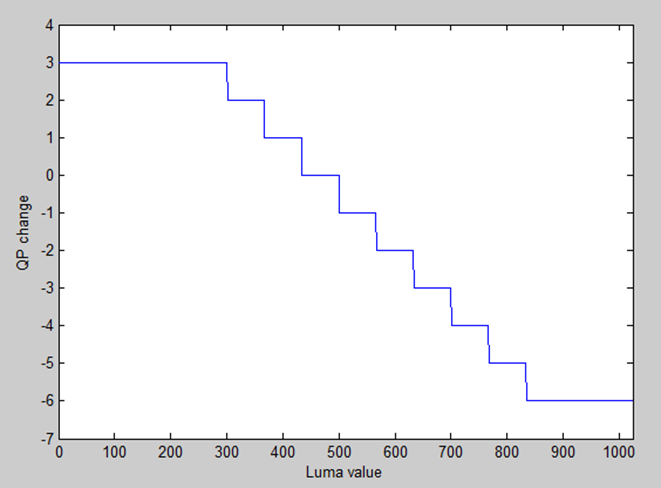


Figure . Change in QP value as a function of the average luma value in the 64x64 pixel block.

# Anchor generation process for ICTCP 420 video

## Software versions

### Conversion software HDRConvert

HDRTools software as specified in section 4.1.1 is used. The conversion process and configuration files for different input HDR content format are as described in section 5.2.

### HDR anchor coding process

The HDR anchors are based on the HEVC Main 10 profile, which implies encodings using 4:2:0 subsampling and 10 bits per channel. The ICTCP colour space is used.

The encoding is performed with HM.

The HM must be configured in 4:2:0 with the following settings:

* The macro RExt\_\_HIGH\_BIT\_DEPTH\_SUPPORT is set to 1, which results in
  + FULL\_NBIT set to 1
  + RExt\_\_HIGH\_PRECISION\_FORWARD\_TRANSFORM set to 1
* Using Random Access (RA) configuration from HEVC common test conditions.

The following example configuration files are provided in the “cfg/cfg\_HDR” folder of the HM reference software package:

* encoder\_randomaccess\_main10\_HDR\_ICtCp\_classAGH.cfg and
* encoder\_randomaccess\_main10\_HDR\_ICtCp\_classBCD.cfg

## Conversion and inverse conversion processes

### RGB linear-light format with BT.2100 primaries for HDR anchor

If the input is in a half float 4:4:4 RGB linear-light format (e.g. OpenEXR), the bit streams shall be generated using the coding / decoding chain illustrated in Figure 6.

The conversion to 4:2:0 10 bits ICTCP is obtained with the HDRConvert tool using the configuration file HDRConvertEXR2020ToICtCp420.cfg provided in the “bin/JCTVC\_CTC\_cfgFiles” folder of the HDRTools reference software package.

This conversion consists of the following steps:

* Convert half precision floating point data to single precision floating point data (not illustrated).
* Convert from RGB (BT.2100) to LMS by invoking section 5.3.1.
* Map using the PQ transfer function (PQ-TF) from LMS (float) to L′M′S′ (float) by invoking section 5.3.3.
* Convert from L′M′S′ (float) to ICTCP by invoking section 5.3.4.
* Quantize from ICTCP (float) into DIDCtDCp (10bit) by invoking section 5.4, with BitDepthY and BitDepthC set to 10.
* Downsample both chroma components from 4:4:4 DIDCtDCp (10bit) to 4:2:0 DIDCtDCp (10bit) by invoking Section 5.5.

Note: Dxx stands for Digitized version of signal xx. L′M′S′ or ICTCP stands for normalized value within [0,1] (R′G′B′ *I*) and [−0.5, 0.5] (*CTCP*).

The reverse conversion is obtained with the HDRConvert tool using the configuration file HDRConvertICtCp420ToEXR2020.cfg provided in the “bin/JCTVC\_CTC\_cfgFiles” folder of the HDRTools reference software package.

The process consists of the following steps:

* Upsample both chroma components from 4:2:0 DIDCtDCp (10bit) to 4:4:4 DIDCtDCp (10bit) by invoking section 5.6.
* Inverse quantize from DIDCtDCp (10bit) into ICTCP (float) by invoking section 5.7, with BitDepthY and BitDepthC set to 10.
* Convert from ICTCP (float) to L′M′S′ (float) by invoking section 5.8.1.
* Inverse map using the inverse PQ-TF from L′M′S′ (float) to LMS (float) by invoking Section 5.8.2.
* Convert from LMS (float) to RGB (float) by invoking section 5.8.3.
* Convert, if needed, the data from single precision floating point numbers to half precision floating point numbers using appropriate rounding operations (not illustrated).



Figure 6. Simplified encoding / decoding chains when input HDR video is RGB linear light.

### Input 4:4:4 DR′DG′DB′ PQ-TF 12bit for HDR Anchor

If the input is using a 4:4:4 PQ-TF 12bit DR′DG′DB′ format, the bit streams shall be generated using the coding / decoding chain illustrated in Figure 7.

#### Input 4:4:4 DR′DG′DB′ PQ-TF 12bit in BT.2100 container

The conversion to 4:2:0 10 bits ICTCP is obtained with the HDRConvert tool using the configuration file HDRConvertBT2020TiffToICtCp420.cfg provided in the “bin/JCTVC\_CTC\_cfgFiles” folder of the HDRTools reference software package.

The process consists of the following steps:

* Convert from 12bit DR′DG′DB′ into normalized R′G′B′ by invoking section 4.12.1.
* Convert from R′G′B′ (BT.2100) to linear RGB by applying the inverse transfer function (inversePQ-TF) as described in Section 4.9.3.
* Convert RGB (BT.2100) to LMS by invoking section 5.3.1.
* Map using the PQ transfer function (PQ-TF) from LMS (float) to L′M′S′ (float) by invoking section 5.3.3.
* Convert from L′M′S′ (float) to ICTCP by invoking section 5.3.4.
* Quantize from ICTCP (float) into DIDCtDCp (10bit) by invoking section 5.4, with BitDepthY and BitDepthC set to 10.
* Downsample both chroma components from 4:4:4 DIDCtDCp (10bit) to 4:2:0 DIDCtDCp (10bit) by invoking Section 5.5.

The reverse conversion is obtained by applying the following steps and the output is DR′DG′DB′ PQ-TF 12b in BT.2100 container. It can be obtained with the HDRConvert tool using the configuration file HDRConvertICtCp420ToBT2020Tiff.cfg provided in the “bin/JCTVC\_CTC\_cfgFiles” folder of the HDRTools reference software package.

* Upsample both chroma components from 4:2:0 DIDCtDCp (10bit) to 4:4:4 DIDCtDCp (10bit) by invoking section 5.6.
* Inverse quantize from DIDCtDCp (10bit) into ICTCP (float) by invoking section 5.7, with BitDepthY and BitDepthC set to 10.
* Convert from ICTCP (float) to L′M′S′ (float) by invoking section 5.8.1.
* Inverse map using the inverse PQ-TF from L′M′S′ (float) to LMS (float) by invoking Section 5.8.2.
* Convert from LMS (float) to RGB (float) by invoking section 5.8.3.
* Map using the PQ transfer function (PQ-TF) from RGB (float) to R′G′B′ (float) by invoking Section 4.4.1.
* Quantize normalized R′G′B′ into 12bit DR′DG′DB′ in BT.2100 container by invoking section 4.12.2.



Figure 7. Simplified encoding / decoding chains when the input HDR video is in 4:4:4 R′G′B′ PQ-TF 12bit.

#### Input 4:4:4 DR′DG′DB′ PQ-TF 12bit in P3D65 container

The conversion to 4:2:0 10 bits ICTCP is obtained with the HDRConvert tool using the configuration file HDRConvertP3D65TiffToICtCp420.cfg provided in the “bin/JCTVC\_CTC\_cfgFiles” folder of the HDRTools reference software package. It consists of the following steps:

* Convert from PQ-TF P3D65 12bit DR′DG′DB′ into normalized R′G′B′ by invoking section 4.12.1.
* Convert from R′G′B′ PQ-TF P3D65 to linear RGB P3D65 by applying the inverse transfer function (inversePQ-TF) as described in section 4.9.3.
* Convert RGB P3D65 to LMS by invoking section 5.3.2.
* Map using the PQ transfer function (PQ-TF) from LMS (float) to L′M′S′ (float) by invoking Section 5.3.3.
* Convert from L′M′S′ (float) to ICTCP by invoking section 5.3.4.
* Quantize from ICTCP (float) into DIDCtDCp (10bit) by invoking Section 5.4, with BitDepthY and BitDepthC set to 10.
* Downsample both chroma components from 4:4:4 DIDCtDCp (10bit) to 4:2:0 DIDCtDCp (10bit) by invoking Section 5.5.

## Colour transformation from RGB to ICTCP

### Conversion from BT.2100 RGB to LMS

* + L = (1688.0 \* R2020 + 2146.0 \* G2020 + 262.0 \* B2020) / 4096.0
  + M = (683.0 \* R2020 + 2951.0 \* G2020 + 462.0 \* B2020) / 4096.0
  + S = (99.0 \* R2020 + 309.0 \* G2020 + 3688.0 \* B2020) / 4096.0

### Conversion from P3D65 RGB to LMS

* + L = 0.334550623220059 \* RP3 + 0.576391094387108 \* GP3 + 0.089058282392833 \* BP3
  + M = 0.158520235234873 \* RP3 + 0.713612932646569 \* GP3 + 0.127866832118557 \* BP3
  + S = 0.020581197388829 \* RP3 + 0.091695662564561 \* GP3 + 0.88772314004661 \* BP3

### Conversion from LMS to L′M′S′

* + L′ = PQ\_TF(max(0, min(L/10000,1)) )
  + M′ = PQ\_TF(max(0, min(M/10000,1)) )
  + S′ = PQ\_TF(max(0, min(S/10000,1)) )

with

### Conversion from L′M′S′ to ICTCP

* + I = (2048.0 \* L′ + 2048.0 \* M′) / 4096.0
  + CT = (6610.0 \* L′ − 13613.0 \* M′ + 7003.0 \* S′) / 4096.0
  + CP = (17933.0 \* L′ − 17390.0 \* M′ – 543.0 \* S′) / 4096.0

## Quantization from ICTCP into DIDCtDCp

This process quantizes the input ICTCP signal into a signal of bit-depth BitDepthY for the I component and BitDepthC for the chroma components (Ct, Cp).

with

Round( x ) = Sign( x ) \* Floor( Abs( x ) + 0.5 )

Sign ( x ) = -1 if x < 0, 0 if x=0, 1 if x > 0

Floor( x ) the largest integer less than or equal to x

Abs( x ) = x if x>=0, -x if x<0

Clip1Y( x ) = Clip3( 0, ( 1 << BitDepthY ) − 1, x )

Clip1C( x ) = Clip3( 0, ( 1 << BitDepthC ) − 1, x )

Clip3( x,y,z ) = x if z<x, y if z>y, z otherwise

## Chroma downsampling from 4:4:4 to 4:2:0

The chroma samples alignment is as follows:



|  |  |  |
| --- | --- | --- |
| Phase k | Coefs c1[k]  4:4:4 🡪 4:2:2  (phase=0) | Coefs c2[k]  4:2:2 🡪 4:2:0  (phase=0) |
| −1 | 1 | 1 |
| 0 | 6 | 6 |
| 1 | 1 | 1 |

* Define shift = 6 and offset = 32.
* Let *H* and *W* be the input picture height and width in chroma samples. For *i*= 0..*H*-1, *j*= 0..*W*/2-1, the intermediate samples *f*[ *i*][ *j*] are derived from the input samples *s*[ *i*][ *j*] as follows:

with Clip3*( x,y,z ) = x* if *z<x,* y if *z>y, z* otherwise

* For *i*= 0..*H*/2-1, *j* = 0..*W*/2-1, the output samples *r*[ *i*][ *j*] are derived from the intermediate samples *f*[ *i*][ *j*] as follows:

*Comment: new filters could be used following first anchors generation steps*

## Chroma upsampling from 4:2:0 to 4:4:4

The upsampling filter used is the same for both horizontal and vertical processes. First, vertical filtering is applied on the 4:2:0 picture, then horizontal filtering.

Filter coefficients values are as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Phase | −2 | −1 | 0 | 1 |
| Coef c[k] | −4 | 36 | 36 | −4 |

Define shift1 = 6, offset1 = 32, shift2 = 12, offset2 = 2048.

Let H and W be the input picture height and width in chroma samples. For i = 0..H−1, j = 0..W−1, the intermediate samples f[ i ][ j ] are derived from the input samples s[ i ][ j ] as follows:

For i = 0..2\*H−1, j = 0..W−1, the output samples r[ i ][ j ] are derived from the intermediate samples f[ i ][ j ] as follows:

*Comment: new filters could be used following first anchors generation steps*

## Inverse Quantization from DIDCtDCp into ICTCP

This process dequantizes the input signal represented on BitDepthY bits for the I component and BitDepthC bits for the chroma components (Ct, Cp) into a (float) signal ICTCP.

with

ClipY(x) = Clip3( 0, 1.0, x)

ClipC(x) = Clip3( −0.5, 0.5, x)

Clip3( x,y,z ) = x if z<x, y if z>y, z otherwise

## Colour transformation from ICTCP to RGB

### ICTCP to L′M′S′

* L′ = clipRGB(I + 0.008609037037933 \* Ct + 0.111029625003026 \* Cp)
* M′ = clipRGB(I − 0.008609037037933 \* Ct − 0.111029625003026 \* Cp)
* S′ = clipRGB(I + 0.560031335710679 \* Ct − 0.320627174987319 \* Cp)

### Conversion from L′M′S′ to LMS

* + L = 10000\*inversePQ\_TF(L′)
  + M = 10000\*inversePQ\_TF(M′)
  + S = 10000\*inversePQ\_TF(S′)

with

### LMS to BT.2100 RGB

* + R = clipRGB(3.436606694333079 \* L − 2.506452118656270 \* M + 0.069845424323191 \* S)
  + G = clipRGB(−0.791329555598929 \* L + 1.983600451792291 \* M − 0.192270896193362 \* S)
  + B = clipRGB(−0.025949899690593 \* L − 0.098913714711726\* M + 1.124863614402319\* S)

with clipRGB( x ) = Clip3( 0, 1, x )

Clip3( x,y,z ) = x if z<x, y if z>y, z otherwise

## HM encoding

The same HM encoding method, as described in section 4.13 is used. Compared to Y′CbCr anchor, the configuration parameters for luma delta QP is kept the same. The configuration parameters for chroma qp offset is listed as in Table 4.

Table . Chroma QP offset parameters for ICTCP PQ signal

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Type** | **WCGPPSChromaQpScale** | **WCGPPSChromaQpOffset** | **WCGPPSCbQpScale** | **WCGPPSCrQpScale** | **CbQPOffset** | **CrQPOffset** |
| **BT.709** | −0.46 | 10 | 0.9 | 1.6 | 6 | 6 |
| **P3** | −0.46 | 10 | 0.5 | 0.9 | 8 | 7 |

Another change of configuration parameters in HM encoder is the MatrixCoefficients setting in the VUI, as shown in Table 5.

Table . VUI parameters in HM encoding

|  |  |
| --- | --- |
|  | **ICTCP PQ** |
| **MatrixCoefficients** | 14 |

1. Since standard range is used, where 0.0 is mapped to code word 64 and 1.0 is mapped to code word 940, the starting interval is [64, 940] rather than [0, 1023]. [↑](#footnote-ref-1)