

**ISO/IEC JTC 1/SC 29/WG 11 N 18911**

**ISO/IEC JTC 1/SC 29/WG 11**

**Coding of moving pictures and audio**

**Convenorship: UNI (Italy)**

|  |  |
| --- | --- |
| Document type: | Approved WG 11 document |
|  |  |
| Title: | Defect report for AVC (ISO/IEC 14496-10) |
|  |  |
| Status: | Approved |
|  |  |
| Date of document: | 2019-10-11 |
|  |  |
| Source: | WG 11 (via JCT-VC) |
|  |  |
| Expected action: | Study |
|  |  |
| No. of pages: | 9 |
|  |  |
| Email of convenor: | leonardo@chiariglione.org |
|  |  |
| Committee URL: | https://isotc.iso.org/livelink/livelink/open/jtc1sc29wg11 |

# General aspects and issues affecting multiple standards

## General and minor matters

Please note that the alignment between the text published by ITU-T and that published by ISO/IEC should also be checked. Minor editorial issues and aspects that are highlighted for potential further checking include the following:

* Updating the reference to Rec. ITU-R BT.2100 (the current version being BT.2100-2) and the associated equations for the ICTCP matrix coefficients interpretation for HLG (esp. check Video CICP; this is correct in Rec. ITU-T H.265 2019-06).

## sYCC colour indicator interpretation

### Status

This item has been confirmed by the JCT-VC and resolved for the ITU-T HEVC text, but has not yet been resolved for the ISO/IEC text. It affects multiple standards: HEVC, AVC, and Video CICP (and JPEG XR). For background, see [JCTVC-AJ0023](http://phenix.it-sudparis.eu/jct/doc_end_user/current_document.php?id=10941).

### Description of the issue

This issue relates to the transfer characteristics and matrix coefficients indicators for the sYCC colour representation specified in IEC 61966-2-1. The the semantics of transfer characteristics (Table E.4 of HEVC), and matrix coefficients (Table E.5 of HEVC) need correction to address the issue.

The issue is a bit complicated because the same transfer characteristics indicator value is used for both sRGB and sYCC, but IEC 61966-2-1 actually indicates that the transfer characteristics function should be somewhat different for the two cases. In the sRGB case, the range of the input value is constrained to be from 0 to 1, but in the sYCC case, this constraint should not apply.

The agreed correction for this aspect is to condition the interpretation of the transfer\_characteristics syntax element for the value 13 on the value of the matrix\_coeffs syntax element (since that value would differ between sYCC and sRGB).

Additionally, the the informative remark relating to the matrix coefficients indicator value for IEC 61966-2-1 sYCC should be changed to indicate that sYCC should be indicated with the matrix coefficients indicator equal to 5 (as for Rec. ITU R BT.601) rather than 1 (as for Rec. ITU-R BT.709).

### Description of the proposed fix

*In E.3.1 (VUI semantics), in Table E.4 (Transfer characteristics interpretation using the transfer\_characteristics syntax element) replace the row for the value 13 with:*

|  |  |  |
| --- | --- | --- |
| 13 | – If matrix\_coeffs is equal to 0           V = *α* \* Lc( 1 ÷ 2.4 ) − ( *α* − 1 ) for 1 >= Lc >= *β*           V = 12.92 \* Lc for *β* > Lc >= 0  – Otherwise           V = *α* \* Lc( 1 ÷ 2.4 ) − ( *α* − 1 ) for Lc >= *β*           V = 12.92 \* Lc for *β* > Lc > −*β*           V = − *α* \* ( −Lc )( 1 ÷ 2.4 ) + ( *α* − 1 ) for −*β* >= Lc | IEC 61966-2-1 sRGB (with matrix\_coeffs equal to 0)  IEC 61966-2-1 sYCC (with matrix\_coeffs equal to 5) |

*In E.3.1 (VUI semantics), after Equation E-3, replace the next paragraph and associated bullet points with:*

In this case, the range of E′R, E′G, and E′B is specified as follows:

– If transfer\_characteristics is equal to 11 or 12, or transfer\_characteristics is equal to 13 and matrix\_coeffs is not equal to 0, E′R, E′G, and E′B are real numbers with values that have a larger range than the range of 0 to 1, inclusive, and their range is not specified in this Specification.

– Otherwise, E′R, E′G and E′B are real numbers in the range of 0 to 1.

*In E.3.1 (VUI semantics), in Table E.5 (Matrix coefficients interpretation using the matrix\_coeffs syntax element), move “*IEC 61966-2-1 sYCC*” from the row for the value 1 to the row for the value 5.*

# Reported errata items for AVC

See also section 1.

## Publication status background

Rec. ITU-T H.264

* (06/19, Edition 13) Approved 2019-06-13, published 2019-09-06

ISO/IEC 14496-10

* ISO/IEC 14496-10:2014 (Edition 8), published 2014-09
* ISO/IEC 14496-10:2014/Amd 1:2015 (Multi-resolution frame compatible stereoscopic video with depth maps, additional supplemental enhancement information and video usability information), published 2015-11
* ISO/IEC 14496-10:2014/FDAMD 2 (Additional Levels and Supplemental Enhancement Information); stage 50.98, deleted in preparation for Edition 9
* ISO/IEC 14496-10:2014/Amd 3:2016 (Additional supplemental enhancement information); published 2016-12, published 2016-12
* ISO/IEC DIS 14496-10:201x (Edition 9); stage 40.99

## Text bug fixes for Annexes I and J

### Status

These bugs were confirmed, and the text bug fixes were agreed by the JCT-VC at its 37th meeting in Geneva in Oct. 2019. See Section 1 of [JCTVC-AK0022](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=10962).

### Bug fixes

*In I.13.2.3.1, change the semantics of da\_mantissa\_len\_minus1 to the following:*

**da\_mantissa\_len\_minus1** + 1 specifies the number of bits in the da\_mantissa syntax element. The value of da\_mantissa\_len\_minus1 shall be in the range of 0 to 31, inclusive. The variable OutManLen is set equal to da\_mantissa\_len\_minus1 + 1.

*In J.7.3.2.13.1, change the depth ranges syntax to the following:*

|  |  |  |
| --- | --- | --- |
| depth\_ranges( numViews, predDirection, index ) { | C | Descriptor |
| **z\_near\_flag** | 11 | u(1) |
| **z\_far\_flag** | 11 | u(1) |
| if( z\_near\_flag ) |  |  |
| 3dv\_acquisition\_element( numViews, predDirection, 7, index, ZNearSign,  ZNearExp, ZNearMantissa, ZNearManLen ) |  |  |
| if( z\_far\_flag ) |  |  |
| 3dv\_acquisition\_element( numViews, predDirection, 7, index, ZFarSign,  ZFarExp, ZFarMantissa, ZFarManLen ) |  |  |
| } |  |  |

*In J.7.3.2.13.2, change the 3DV acquisition element syntax as follows:*

|  |  |  |
| --- | --- | --- |
| 3dv\_acquisition\_element( numViews, predDirection, expLen, index, outSign, outExp, outMantissa, outManLen ) { | **C** | Descriptor |
| if( numViews > 1 ) |  |  |
| **element\_equal\_flag** | 11 | u(1) |
| if( element\_equal\_flag  = =  0 ) |  |  |
| numValues = numViews |  |  |
| else |  |  |
| numValues = 1 |  |  |
| for( i = 0; i < numValues; i++ ) { |  |  |
| if( predDirection  = =  2  &&  i  = =  0 ) { |  |  |
| **mantissa\_len\_minus1** | 11 | u(5) |
| outManLen[ index, i ] = manLen = mantissa\_len\_minus1 + 1 |  |  |
| } |  |  |
| if( predDirection  = =  2 ) { |  |  |
| **sign0** | 11 | u(1) |
| outSign[ index, i ] = sign0 |  |  |
| **exponent0** | 11 | u(v) |
| outExp[ index, i ] = exponent0 |  |  |
| **mantissa0** | 11 | u(v) |
| outMantissa[ index, i ] = mantissa0 |  |  |
| } else { |  |  |
| **skip\_flag** | 11 | u(1) |
| if( skip\_flag = = 0 ) { |  |  |
| **sign1** | 11 | u(1) |
| outSign[ index, i ] = sign1 |  |  |
| **exponent\_skip\_flag** | 11 | u(1) |
| if( exponent\_skip\_flag = = 0 ) { |  |  |
| **exponent1** | 11 | u(v) |
| outExp[ index, i ] = exponent1 |  |  |
| } else |  |  |
| outExp[ index, i ] = outExp[ ref\_dps\_id0, i ] |  |  |
| **mantissa\_diff** | 11 | se(v) |
| if( predDirection = = 0 ) |  |  |
| mantissaPred = (( outMantissa[ ref\_dps\_id0, i ] \* predWeight0 +  outMantissa[ ref\_dps\_id1, i ] \* ( 64-predWeight0 ) + 32 ) >> 6 ) |  |  |
| else |  |  |
| mantissaPred = outMantissa[ ref\_dps\_id0, i ] |  |  |
| outMantissa[ index, i ] = mantissaPred + mantissa\_diff |  |  |
| outManLen[ index, i ] = outManLen[ ref\_dps\_id0, i ] |  |  |
| } else { |  |  |
| outSign[ index, i ] = outSign[ ref\_dps\_id0, i ] |  |  |
| outExp[ index, i ] = outExp[ ref\_dps\_id0, i ] |  |  |
| outMantissa[ index, i ] = outMantissa[ ref\_dps\_id0, i ] |  |  |
| outManLen[ index, i ] = outManLen[ ref\_dps\_id0, i ] |  |  |
| } |  |  |
| } |  |  |
| } |  |  |
| if( element\_equal\_flag = = 1 ) { |  |  |
| for( i = 1; i < numViews; i++ ) { |  |  |
| outSign[ index, i ] = outSign[ index, 0 ] |  |  |
| outExp[ index, i ] = outExp[ index, 0 ] |  |  |
| outMantissa[ index, i ] = outMantissa[ index, 0 ] |  |  |
| outManLen[ index, i ] = outManLen[ index, 0 ] |  |  |
| } |  |  |
| } |  |  |
| } |  |  |

*In J.7.4.2.13.2, change the 3DV acquisition element semantics as follows:*

The syntax structure specifies the value of an element in the depth ranges syntax structure. The element may contain one or more loop entries i of the order specified by view\_id\_3dv syntax elements.

The contents of the syntax structure are controlled through input variables predDirection, expLen, and index, the semantics of which are as follows.

– predDirection equal to 2 specifies that the first loop entry of the element is not predicted and coded in the sign, exponent, and mantissa syntax elements. predDirection equal to 0 or 1 specifies that the first loop entry of the element is predicted and a difference relative to a prediction value is coded in the difference syntax element.

– expLen specifies the number of bits in the exponent syntax element.

– index greater than 0 specifices the depth\_parameter\_set\_id of the depth parameter set wherein the parameters are present, and index equal to 0 specifies that the parameters are present in a sequence parameter set.

The syntax structure uses outSign, outExp, outMantissa, and outManLen variables for both input and output, where each variable is indexed by [ index, viewIdc ], index being an identifier (equal to either 0 when decoding depth ranges in sequence parameter set or depth\_parameter\_set\_id value when decoding depth range parameter set) to a depth parameter set and viewIdc being a view indicator (in the order of views for 3DV acquisition parameters).

**element\_equal\_flag** equal to 0 specifies that the sign, exponent, and mantissa may or may not be identical to respective values for any two loop entries i and j. element\_equal\_flag equal to 1 specifies that the sign, exponent, and mantissa are identical to respective values for any two loop entries i and j. When not present, element\_equal\_flag is inferred to be equal to 0.

**mantissa\_len\_minus1** plus 1 specifies the number of bits in the mantissa syntax element. The value of mantissa\_len\_minus1 shall be in the range of 0 to 31, inclusive.

**sign0** equal to 0 indicates that the sign of the value provided in the loop entry is positive. sign0 equal to 1 indicates that the sign is negative.

**exponent0** specifies the exponent of the value provided by the loop entry. The syntax element exponent0 is represented by expLen bits. The value of exponent0 shall be in the range of 0 to 2expLen – 2, inclusive. The value 2expLen – 1 is reserved for future use by ITU‑T | ISO/IEC. Decoders shall treat the value 2expLen – 1 as indicating an unspecified value.

**mantissa0** specifies the mantissa of the value provided by the loop entry. The syntax element mantissa0 is represented by manLen bits.

**skip\_flag** equal to 0 specifies that syntax elements sign1, exponent\_skip\_flag and mantissa\_diff are present for the loop entry. skip\_flag equal to 1 specifies that elements sign1, exponent\_skip\_flag and mantissa\_diff are not present for the loop entry.

**sign1** equal to 0 indicates that the sign of the value provided in the loop entry is positive. sign1 equal to 1 indicates that the sign is negative.

**exponent1**, if present, specifies the exponent of the value provided by the loop entry. The syntax element exponent1 is represented by expLen bits. The value of exponent1 shall be in the range of 0 to 2expLen – 2, inclusive. The value 2expLen – 1 is reserved for future use by ITU‑T | ISO/IEC. Decoders shall treat the value 2expLen – 1 as indicating an unspecified value.

**mantissa\_diff** specifies the difference of the mantissa of the value provided by the loop entry relative to its prediction value.

## On semantics of nal\_hrd\_parameters\_present\_flag and vcl\_hrd\_parameters\_present\_flag

### Status

These bugs were confirmed, and the text bug fixes were agreed by the JCT-VC at its 37th meeting in Geneva in Oct. 2019. See Section 2 of [JCTVC-AK0022](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=10962).

### Bug fixes

*Change the semantics of nal\_hrd\_parameters\_present\_flag and vcl\_hrd\_parameters\_present\_flag as follows (additions are yellow-highlighted):*

**nal\_hrd\_parameters\_present\_flag** equal to 1 specifies that NAL HRD parameters (pertaining to the Type II bitstream conformance point) are present. nal\_hrd\_parameters\_present\_flag equal to 0 specifies that NAL HRD parameters are not present.

NOTE 12 – When nal\_hrd\_parameters\_present\_flag is equal to 0, the conformance of the bitstream cannot be verified without provision of the NAL HRD parameters and all buffering period SEI messages, and, when vcl\_hrd\_parameters\_present\_flag is also equal to 0, all picture timing SEI messages, by some means not specified in this Recommendation | International Standard.

...

**vcl\_hrd\_parameters\_present\_flag** equal to 1 specifies that VCL HRD parameters (pertaining to the Type I ~~all~~ bitstream conformance point) are present. vcl\_hrd\_parameters\_present\_flag equal to 0 specifies that VCL HRD parameters are not present.

NOTE 13 – When vcl\_hrd\_parameters\_present\_flag is equal to 0, the conformance of the bitstream cannot be verified without provision of the VCL HRD parameters and all buffering period SEI messages, and, when nal\_hrd\_parameters\_present\_flag is also equal to 0, all picture timing SEI messages, by some means not specified in this Recommendation | International Standard.

...

## On semantics of rbsp\_byte[ i ]

### Status

These bugs were confirmed, and the text bug fixes were agreed by the JCT-VC at its 37th meeting in Geneva in Oct. 2019. See [JCTVC-AK0023](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=10963).

### Bug fixes

*Change the semantics of rbsp\_byte[ i ] to the following:*

**rbsp\_byte**[ i ] is the i-th byte of an RBSP. An RBSP is specified as an ordered sequence of bytes as follows:

The RBSP contains a string of data bits (SODB) as follows:

– If the SODB is empty (i.e., zero bits in length), the RBSP is also empty.

– Otherwise, the RBSP contains the SODB as follows:

1) The first byte of the RBSP contains the first (most significant, left-most) eight bits of the SODB; the next byte of the RBSP contains the next eight bits of the SODB, etc., until fewer than eight bits of the SODB remain.

2) The rbsp\_trailing\_bits( ) syntax structure is present after the SODB as follows:

i) The first (most significant, left-most) bits of the final RBSP byte contain the remaining bits of the SODB (if any).

ii) The next bit consists of a single bit equal to 1 (i.e., rbsp\_stop\_one\_bit).

iii) When the rbsp\_stop\_one\_bit is not the last bit of a byte-aligned byte, one or more zero-valued bits (i.e., instances of rbsp\_alignment\_zero\_bit) are present to result in byte alignment.

3) One or more cabac\_zero\_word 16-bit syntax elements equal to 0x0000 may be present in some RBSPs after the rbsp\_trailing\_bits( ) at the end of the RBSP.