|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  37th Meeting: Geneva, CH, 4–10 October 2019 | Document: JCTVC-AK0022-v1 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Some AVC errata items** | | |
| *Status:* | Input document to JCT-VC | | |
| *Purpose:* | Editorial input | | |
| *Author(s) or Contact(s):* | **Miska M. Hannuksela** Nokia, Tampere, Finland  **Dong Tian** InterDigital, Princeton, NJ, USA  **Ye-Kui Wang** Futurewei, San Diego, CA, USA | Tel: Email: | [miska.hannuksela@nokia.com](mailto:miska.hannuksela@nokia.com)  [dong.tian@interdigital.com](mailto:dong.tian@interdigital.com)  [ye-kui.wang@futurewei.com](mailto:ye-kui.wang@futurewei.com) |
| *Source:* | AHG2 | | |

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

# Abstract

This contribution reports some text bugs and proposes corresponding fixes for the AVC specifications.

# Proposed text bug fixes for Annexes I and J

*In I.13.2.3.1, change the semantics of da\_mantissa\_len\_minus1as follows (with change marks):*

**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 as follows (with change marks):*

|  |  |  |
| --- | --- | --- |
| 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

*Change the semantics of nal\_hrd\_parameters\_present\_flag and vcl\_hrd\_parameters\_present\_flag as follows (additions are yellow-highlighted, removals are strikethrough in red fonts):*

**nal\_hrd\_parameters\_present\_flag** equal to 1 specifies that NAL HRD parameters (pertaining to Type II bitstream conformance) 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 Type II 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 the 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 Type I ~~all~~ bitstream conformance) 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 Type I conformance of the bitstream cannot be verified without provision of the VCL HRD parameters and all buffering period SEI messages, and, when vcl\_hrd\_parameters\_present\_flag is also equal to 0, all the picture timing SEI messages, by some means not specified in this Recommendation | International Standard.