|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11**  7th Meeting: Geneva, CH, 21–30 November, 2011 | Document: JCTVC-G1103\_d9 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **WD5: Working Draft 5 of High-Efficiency Video Coding** | | |
| *Status:* | Output Document of JCT-VC | | |
| *Purpose:* | Working Draft of HEVC | | |
| *Author(s) or Contact(s):* | Benjamin Bross Fraunhofer HHI  Woo-Jin Han Gachon University  Jens-Rainer Ohm RWTH Aachen  Gary J. Sullivan Microsoft  Thomas Wiegand Fraunhofer HHI / TU Berlin | Email:  Email:  Email:  Email:  Email: | benjamin.bross@hhi.fraunhofer.de  hurumi@gmail.com  ohm@ient.rwth-aachen.de  garysull@microsoft.com  thomas.wiegand@hhi.fraunhofer.de |
| *Source:* | Editor | | |

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

# Abstract

Changes to relevant text in Working Draft 5 of High-Efficiency Video Coding to incoporate JCTVC-H0531.

### Slice header syntax

|  |  |  |  |
| --- | --- | --- | --- |
| slice\_header( ) { | | Descriptor | |
| **first\_slice\_in\_pic\_flag** | | u(1) | |
| if( first\_slice\_in\_pic\_flag = = 0 ) | |  | |
| **slice\_address** | | u(v) | |
| **slice\_type** | | ue(v) | |
| **entropy\_slice\_flag** | | u(1) | |
| if( !entropy\_slice\_flag ) { | |  | |
| **pic\_parameter\_set\_id** | | ue(v) | |
| if( IdrPicFlag ) { | |  | |
| **idr\_pic\_id** | | ue(v) | |
| **no\_output\_of\_prior\_pics\_flag** | | u(1) | |
| } | |  | |
| else { | |  | |
| **pic\_order\_cnt\_lsb** | | u(v) | |
| **short\_term\_ref\_pic\_set\_pps\_flag** | | u(1) | |
| if( !short\_term\_ref\_pic\_set\_pps\_flag ) | |  | |
| short\_term\_ref\_pic\_set( num\_short\_term\_ref\_pic\_sets ) | |  | |
| Else | |  | |
| **short\_term\_ref\_pic\_set\_idx** | | u(v) | |
| if( long\_term\_ref\_pics\_present\_flag ) { | |  | |
| **num\_long\_term\_pics** | | ue(v) | |
| for( i = 0; i < num\_long\_term\_pics; i++ ) { | |  | |
| **delta\_poc\_lsb\_lt** [ i ] | | ue(v) | |
| **delta\_poc\_msb\_present\_flag**[ i ] | | u(1) | |
| if( delta\_poc\_msb\_present\_flag[ i ] = = 1 ) | |  | |
| **delta\_poc\_msb\_cycle\_lt\_minus1**[ i ] | | ue(v) | |
| **used\_by\_curr\_pic\_lt\_flag**[ i ] | | u(1) | |
| } | |  | |
| } | |  | |
| } | |  | |
| if( scaling\_list\_enable\_flag | |   deblocking\_filter\_in\_APS\_enabled\_flag | |   sample\_adaptive\_offset\_enabled\_flag | |   adaptive\_loop\_filter\_enabled\_flag ) { | |  | |
| if( sample\_adaptive\_offset\_enabled\_flag ) | |  | |
| **Slice\_sample\_adaptive\_offset\_flag** | | u(1) | |
| if( adaptive\_loop\_filter\_enabled\_flag) | |  | |
| **Slice\_adaptive\_loop\_filter\_flag** | | u(1) | |
| **aps\_id** | | ue(v) | |
| } | |  | |
| if( slice\_type = = P | | slice\_type = = B ) { | |  | |
| **Num\_ref\_idx\_active\_override\_flag** | | u(1) | |
| if( num\_ref\_idx\_active\_override\_flag ) { | |  | |
| **Num\_ref\_idx\_l0\_active\_minus1** | | ue(v) | |
| if( slice\_type = = B ) | |  | |
| **num\_ref\_idx\_l1\_active\_minus1** | | ue(v) | |
| } | |  | |
| } | |  | |
| ref\_pic\_list\_modification( ) | |  | |
| ref\_pic\_list\_combination( ) | |  | |
| } | |  | |
| if( slice\_type != I ) | |  | |
| **cabac\_init\_idc** | | ue(v) | |
| if( !entropy\_slice\_flag ) { | |  | |
| **slice\_qp\_delta** | | se(v) | |
| **inherit\_dbl\_params\_from\_APS\_flag** | | u(1) | |
| if ( !inherit\_dbl\_params\_from\_APS\_flag ) { | |  | |
| **Disable\_deblocking\_filter\_flag** | | u(1) | |
| if ( !disable\_deblocking\_filter\_flag ) { | |  | |
| beta\_offset\_div2 | | se(v) | |
| tc\_offset\_div2 | | se(v) | |
| } | |  | |
| } | |  | |
| if( slice\_type = = B ) | |  | |
| **collocated\_from\_l0\_flag** | | u(1) | |
| if( ( weighted\_pred\_flag && slice\_type = = P) | |  ( weighted\_bipred\_idc = = 1 && slice\_type = = B ) ) | |  | |
| pred\_weight\_table( ) | |  | |
| } | |  | |
| if( slice\_type = = P | | slice\_type = = B ) | |  | |
| **5\_minus\_max\_num\_merge\_cand** | | ue(v) | |
| if( adaptive\_loop\_filter\_enabled\_flag && aps\_adaptive\_loop\_filter\_flag ) | |  | |
| alf\_cu\_control\_param( ) | |  | |
| for( i = 0; i <num\_substreams\_minus1 + 1; i++ ){ | |  | |
| **Substream\_length\_mode** | | u(2) | |
| **Substream\_length[i]** | | u(v) | |
| } | |  | |
| } | |  | |

### Slice header semantics

When present, the value of the slice header syntax elements pic\_parameter\_set\_id, idr\_pic\_id, no\_output\_of\_prior\_pics\_flag, pic\_order\_cnt\_lsb, short\_term\_ref\_pic\_set\_pps\_flag, short\_term\_ref\_pic\_set\_idx and num\_long\_term\_pics shall be the same in all slice headers of a coded picture. When present, the value of the slice header syntax elements delta\_poc\_lsb\_lt\_minus1[ i ] and used\_by\_curr\_pic\_lt\_flag[ i ] shall be the same in all slice headers of a coded picture for each i in the range of 0 to num\_long\_term\_pics, inclusive.

**first\_slice\_in\_pic\_flag** indicates whether the slice is the first slice of the picture. If first\_slice\_in\_pic\_flag is equal to 1, the variables SliceAddress and LCUAddress are both set to 0 and the decoding starts with the first LCU in the picture.

**slice\_address** specifies the address in slice granularity resolution in which the slice starts and shall be represented by ( Ceil( Log2( NumLCUsInPicture ) ) + SliceGranularity ) bits in the bitstream where NumLCUsInPicture is the number of LCUs in a picture.

The variable LCUAddress is set to ( slice\_address >> SliceGranularity ) and represents the LCU part of the slice address in raster scan order. The variable GranularityAddress is set to ( slice\_address - ( LCUAddress << SliceGranularity ) ) and represents the sub-LCU part of the slice address expressed in z-scan order.

[Ed. (BB): “z-scan order” is not defined anywhere and should be defined, or the phrase herein is changed not to use the term.]

The variable SliceAddress is then set to ( LCUAddress << ( log2\_diff\_max\_min\_coding\_block\_size << 1 ) ) + ( GranularityAddress << ( ( log2\_diff\_max\_min\_coding\_block\_size << 1 ) – SliceGranularity ) and the slice decoding starts with the largest coding unit possible at the slice starting coordinate.

**slice\_type** specifies the coding type of the slice according to Table 7‑3.

Table 07‑13 – Name association to slice\_type

|  |  |
| --- | --- |
| slice\_type | Name of slice\_type |
| 0 | P (P slice) |
| 1 | B (B slice) |
| 2 | I (I slice) |

When nal\_unit\_type is equal to 5 (IDR picture), slice\_type shall be equal to 2.

When max\_num\_ref\_frames is equal to 0, slice\_type shall be equal to 2.

**entropy\_slice\_flag** equal to 1 specifies that the value of slice header syntax elements not present shall be inferred to be equal to the value of slice header syntax elements in a proceeding slice, where a proceeding slice is defined as the slice containing treeblock with location (LCUAddress – 1). entropy\_slice\_flag shall be equal to 0 when LCUAddress equal to 0.

**pic\_parameter\_set\_id** specifies the picture parameter set in use. The value of pic\_parameter\_set\_id shall be in the range of 0 to 255, inclusive.

**idr\_pic\_id** identifies an IDR picture. The values of idr\_pic\_id in all the slices of an IDR picture shall remain unchanged. When two consecutive access units in decoding order are both IDR access units, the value of idr\_pic\_id in the slices of the first such IDR access unit shall differ from the idr\_pic\_id in the second such IDR access unit. The value of idr\_pic\_id shall be in the range of 0 to 65535, inclusive.

**no\_output\_of\_prior\_pics\_flag** specifies how the previously-decoded pictures in the decoded picture buffer are treated after decoding of an IDR picture. See Annex C. When the IDR picture is the first IDR picture in the bitstream, the value of no\_output\_of\_prior\_pics\_flag has no effect on the decoding process. When the IDR picture is not the first IDR picture in the bitstream and the value of pic\_width\_in\_luma\_samples or pic\_height\_in\_luma\_samples or max\_dec\_frame\_buffering derived from the active sequence parameter set is different from the value of pic\_width\_in\_luma\_samples or pic\_height\_in\_luma\_samples or max\_dec\_frame\_buffering derived from the sequence parameter set active for the preceding picture, no\_output\_of\_prior\_pics\_flag equal to 1 may (but should not) be inferred by the decoder, regardless of the actual value of no\_output\_of\_prior\_pics\_flag.

**pic\_order\_cnt\_lsb** specifies the picture order count modulo MaxPicOrderCntLsb for the current picture. The length of the pic\_order\_cnt\_lsb syntax element is log2\_max\_pic\_order\_cnt\_lsb\_minus4 + 4 bits. The value of the pic\_order\_cnt\_lsb shall be in the range of 0 to MaxPicOrderCntLsb − 1, inclusive. When pic\_order\_cnt\_lsb is not present, pic\_order\_cnt\_lsb shall be inferred to be equal to 0.

**short\_term\_ref\_pic\_set\_pps\_flag** equal to 1 specifies that the short-term reference picture set of the current picture shall be created using syntax elements in the active picture parameter set. short\_term\_ref\_pic\_set\_pps\_flag equal to 0 specifies that the short-term reference picture set of the current picture shall be created using syntax elements in the short\_term\_ref\_pic\_set( ) syntax structure in the slice header.

**short\_term\_ref\_pic\_set\_idx** specifies the index to the list of the short-term reference picture sets specified in the active picture parameter set that shall be used for creation of the reference picture set of the current picture. The syntax element short\_term\_ref\_pic\_set\_idx shall be represented by ceil( log2( num\_short\_term\_ref\_pic\_sets ) ) bits. The value of short\_term\_ref\_pic\_set\_idx shall be in the range of 0 to num\_short\_term\_ref\_pic\_sets – 1, inclusive, where num\_short\_term\_ref\_pic\_sets is the syntax element from the active picture parameter set.

The variable StRpsIdx is derived as follows.

if( short\_term\_ref\_pic\_set\_pps\_flag )  
 StRpsIdx = short\_term\_ref\_pic\_set\_idx (7‑9)  
else  
 StRpsIdx = num\_short\_term\_ref\_pic\_sets

**num\_long\_term\_pics** specifies the number of the long-term reference pictures that are to be included in the long-term reference picture set of the current picture. The value of num\_long\_term\_pics shall be in the range of 0 to max\_num\_ref\_frames – NumNegativePics[ StRpsIdx ] – NumPositivePics[ StRpsIdx ] , inclusive. When not present, the value of num\_long\_term\_pics shall be inferred to be equal to 0.

**delta\_poc\_lsb\_lt** [ i ] is used to determine the value of the least significant bits of the picture order count value of the i-th long-term reference picture that is included in the long-term reference picture set of the current picture. delta\_poc\_lsb\_lt\_minus1[ i ] shall be in the range of 0 to MaxPicOrderCntLsb − 1, inclusive.

The variable DeltaPocLt[ i ] is derived as follows.

if (i = = 0)   
 DeltaPocLt[ i ] = delta\_poc\_lsb\_lt[ i ] (7‑9)  
else  
 DeltaPocLt[ i ] = delta\_poc\_lsb\_lt[ i ] + DeltaPocLt[ i – 1 ]

The value of DeltaPocLt[ i ] shall be in the range of 0 to MaxPicOrderCntLsb, inclusive.

**delta\_poc\_msb\_present\_flag[i]** equal to 1 specifies that delta\_poc\_msb\_cycle\_lt\_minus1[ i ] is present. delta\_poc\_msb\_present\_flag[ i ]equal to 0 specifies that delta\_poc\_msb\_cycle\_lt\_minus1[ i ] is not present. delta\_poc\_msb\_present\_flag[ i ] shall be equal to 1 when DeltaPocLt[ i ] is equal to DeltaPocLt[ j ] for values of j in the range of 0 to num\_long\_term\_pics – 1, inclusive, and being unequal to i.

**delta\_poc\_msb\_cycle\_lt\_minus1**[ i ]is used to determine the value of the most significant bits of the picture order count value of the i-th long-term reference picture that is included in the long-term reference picture set of the current picture.

The variable DeltaPocMSBCycleLt[ i ] is derived as follows.

if ( i = = 0 | | DeltaPocLt[ i – 1 ] ! = DeltaPocLt[ i ] )   
 DeltaPocMSBCycleLt[ i ] = delta\_poc\_msb\_cycle\_lt\_minus1[ i ] + 1   
 else  
 DeltaPocMSBCycleLt[ i ] = delta\_poc\_msb\_cycle\_lt\_minus1[ i ] + 1 +

DeltaPocMSBCycleLt [ i – 1 ]

**used\_by\_curr\_pic\_lt\_flag**[ i ] equal to 0 specifies that the i-th long-term reference picture included in the long-term reference picture set of the current picture is not used for reference by the current picture.

**num\_ref\_idx\_active\_override\_flag** equal to 1 specifies that the syntax element num\_ref\_idx\_l0\_active\_minus1 is present for P and B slices and that the syntax element num\_ref\_idx\_l1\_active\_minus1 is present for B slices. num\_ref\_idx\_active\_override\_flag equal to 0 specifies that the syntax elements num\_ref\_idx\_l0\_active\_minus1 and num\_ref\_idx\_l1\_active\_minus1 are not present.

When the current slice is a P or B slice and field\_pic\_flag is equal to 0 and the value of num\_ref\_idx\_l0\_default\_active\_minus1 in the picture parameter set exceeds 15, num\_ref\_idx\_active\_override\_flag shall be equal to 1.

When the current slice is a B slice and field\_pic\_flag is equal to 0 and the value of num\_ref\_idx\_l1\_default\_active\_minus1 in the picture parameter set exceeds 15, num\_ref\_idx\_active\_override\_flag shall be equal to 1.

**num\_ref\_idx\_l0\_active\_minus1** specifies the maximum reference index for reference picture list 0 that shall be used to decode the slice.

When the current slice is a P or B slice and num\_ref\_idx\_l0\_active\_minus1 is not present, num\_ref\_idx\_l0\_active\_minus1 shall be inferred to be equal to num\_ref\_idx\_l0\_default\_active\_minus1.

The range of num\_ref\_idx\_l0\_active\_minus1 is specified as follows.p

– If field\_pic\_flag is equal to 0, num\_ref\_idx\_l0\_active\_minus1 shall be in the range of 0 to 15, inclusive. When MbaffFrameFlag is equal to 1, num\_ref\_idx\_l0\_active\_minus1 is the maximum index value for the decoding of frame macroblocks and 2 \* num\_ref\_idx\_l0\_active\_minus1 + 1 is the maximum index value for the decoding of field macroblocks.

– Otherwise (field\_pic\_flag is equal to 1), num\_ref\_idx\_l0\_active\_minus1 shall be in the range of 0 to 31, inclusive.

**num\_ref\_idx\_l1\_active\_minus1** specifies the maximum reference index for reference picture list 1 that shall be used to decode the slice.

When the current slice is a B slice and num\_ref\_idx\_l1\_active\_minus1 is not present, num\_ref\_idx\_l1\_active\_minus1 shall be inferred to be equal to num\_ref\_idx\_l1\_default\_active\_minus1.

The range of num\_ref\_idx\_l1\_active\_minus1 is constrained as specified in the semantics for num\_ref\_idx\_l0\_active\_minus1 with l0 and list 0 replaced by l1 and list 1, respectively.

**cabac\_init\_idc** specifies the index for determining the initialisation table used in the initialisation process for context variables. The value of cabac\_init\_idc shall be in the range of 0 to 2, inclusive.

**slice\_qp\_delta** specifies the initial value of QPY to be used for all the macroblocks in the slice until modified by the value of cu\_qp\_delta in the coding unit layer. The initial QPY quantisation parameter for the slice is computed as

SliceQPY = 26 + pic\_init\_qp\_minus26 + slice\_qp\_delta (7‑9)

The value of slice\_qp\_delta shall be limited such that SliceQPY is in the range of −QpBdOffsetY to +51, inclusive.

**disable\_deblocking\_filter\_flag** equal to 1 specifies that the operation of the deblocking filter shall be disabled and disable\_deblocking\_filter\_flag equal to 0 specifies that the operation of the deblocking filter shall be enabled.

**collocated\_from\_l0\_flag** equal to 1 specifies the picture that contains the co-located partition shall be derived from list 0, otherwise the picture shall be derived from list 1.

**5\_minus\_max\_num\_merge\_cand** specifies the maximum number of merging MVP candidates supported in the slice subtracted from 5. The maximum number of merging MVP candidates, MaxNumMergeCand is computed as

MaxNumMergeCand = 5 − 5\_minus\_max\_num\_merge\_cand (7‑9)

The value of 5\_minus\_max\_num\_merge\_cand shall be limited such that MaxNumMergeCand is in the range of 0 to 5, inclusive.

**substream\_length\_mode** specifies the number of bits to be used to decode syntax element substream\_length[ ], as defined in Table 7‑4.

Table ‑ – Number of bits used to decode substream\_length

|  |  |
| --- | --- |
| **substream\_length\_mode** | **Number of bits** |
| 0 | 8 |
| 1 | 16 |
| 2 | 24 |
| 3 | 32 |

**substream\_length[i]** specifies the number of bits to be used during the initialization process for the current bitstream pointer i.

## Slice decoding process

### Decoding process for picture order count

Output of this process is PicOrderCntVal, the picture order count of the current picture.

Picture order counts are used to identify pictures, for deriving motion parameters in temporal or spatial direct mode, to represent picture order differences between frames for motion vector derivation, for implicit mode weighted prediction in B slices (see subclause **Error! Reference source not found.**), and for decoder conformance checking (see subclause C.4).

Each coded picture is associated with one picture order count, called PicOrderCntVal. PicOrderCntVal indicates the picture order of the corresponding picture relative to the previous IDR picture in decoding order.

The variables prevPicOrderCntLsb and prevPicOrderCntMsb are derived as follows.

* If the current picture is an IDR picture, both prevPicOrderCntLsb and prevPicOrderCntMsb are set equal to 0.

– Otherwise (the current picture is not an IDR picture), the following applies.

– Let prevRefPic be the previous reference picture in decoding order that has temporal\_id equal to 0. The variable prevPicOrderCntLsb is set equal to pic\_order\_cnt\_lsb of prevRefPic, and the variable prevPicOrderCntMsb is set equal to PicOrderCntMsb of prevRefPic.

PicOrderCntMsb of the current picture is derived as specified by the following pseudo-code:

if( ( pic\_order\_cnt\_lsb < prevPicOrderCntLsb ) &&  
 ( ( prevPicOrderCntLsb − pic\_order\_cnt\_lsb ) >= ( MaxPicOrderCntLsb / 2 ) ) )  
 PicOrderCntMsb = prevPicOrderCntMsb + MaxPicOrderCntLsb (8‑9)  
else if( (pic\_order\_cnt\_lsb > prevPicOrderCntLsb ) &&  
 ( (pic\_order\_cnt\_lsb − prevPicOrderCntLsb ) > ( MaxPicOrderCntLsb / 2 ) ) )  
 PicOrderCntMsb = prevPicOrderCntMsb − MaxPicOrderCntLsb  
else  
 PicOrderCntMsb = prevPicOrderCntMsb

PicOrderCntVal is derived as

PicOrderCntVal = PicOrderCntMsb + pic\_order\_cnt\_lsb (8‑9)

NOTE 1 – All IDR pictures will have PicOrderCntVal equal to 0 since pic\_order\_lsb is inferred to be 0 for IDR pictures and prevPicOrderCntLsb and prevPicOrderCntMsb are both set equal to 0.

The value of PicOrderCntVal shall be in the range of −231 to 231 − 1, inclusive. In one coded video sequence, the PicOrderCntVal values for any two coded pictures shall be different.

The variable maxPicOrderCnt is set equal to the maximum value of PicOrderCntVal among the associated values for the current picture and all pictures in the DPB that are currently marked as "used for short-term reference" or "needed for output".

The variable minPicOrderCnt is set equal to the minimum value of PicOrderCntVal among the associated values for the current picture and all pictures in the DPB that are currently marked as "used for short-term reference" or "needed for output".

It is a requirement of bitstream conformance that the value of maxPicOrderCnt – minPicOrderCnt shall be less than MaxPicOrderCntLsb / 2.

The function PicOrderCnt( picX ) is specified as follows:

PicOrderCnt( picX ) = PicOrderCntVal of the picture picX (8‑9)

The function DiffPicOrderCnt( picA, picB ) is specified as follows:

DiffPicOrderCnt( picA, picB ) = PicOrderCnt( picA ) − PicOrderCnt( picB ) (8‑9)

The bitstream shall not contain data that result in values of DiffPicOrderCnt( picA, picB ) used in the decoding process that exceed the range of −215 to 215 − 1, inclusive.

NOTE 2 – Let X be the current picture and Y and Z be two other pictures in the same sequence, Y and Z are considered to be in the same output order direction from X when both DiffPicOrderCnt( X, Y ) and DiffPicOrderCnt( X, Z ) are positive or both are negative.

NOTE 3 – Many encoders assign PicOrderCntVal proportional to the sampling time of the corresponding picture relative to the sampling time of the previous IDR picture.

### Decoding process for reference picture set

This process is invoked once per picture, after decoding of a slice header but prior to the decoding of any coding unit and prior to the decoding process for reference picture list construction of the slice as specified in subclause 0.1.4. The process may result in marking one or more reference pictures as "unused for reference".

NOTE 1 – The reference picture set is an absolute description of the reference pictures used in the decoding process of the current and future coded pictures. The reference picture set signaling is explicit in the sense that all reference pictures included in the reference picture set are listed explicitly and there is no default reference picture set construction process in the decoder that depends on the status of the decoded picture buffer.

Short-term reference pictures are identified by their PicOrderCntVal values. Long-term reference pictures are identified by their pic\_order\_cnt\_lsb values.

Five lists of picture order count values are constructed to derive the reference picture set; PocStCurr0, PocStCurr1, PocStFoll, PocLtCurr, and PocLtFoll with NumPocStCurr0, NumPocStCurr1, NumPocStFoll, NumPocLtCurr, and NumPocLtFoll number of elements, respectively.

If the current picture is an IDR picture, PocStCurr0, PocStCurr1, PocStFoll, PocLtCurr, and PocLtFoll are all set to empty, and NumPocStCurr0, NumPocStCurr1, NumPocStFoll, NumPocLtCurr, and NumPocLtFoll are all set to 0.

Otherwise, the following applies for derivation of the five lists of picture order count values and the numbers of entries.

for( i = 0, j = 0, k = 0; i < NumNegativePics[ StRpsIdx ] ; i++ ) {  
 if( UsedByCurrPicS0[ StRpsIdx ][ i ] )  
 PocStCurr0[ j++ ] = PicOrderCntVal + DeltaPocS0[ StRpsIdx ][ i ]  
 else  
 PocStFoll[ k++ ] = PicOrderCntVal + DeltaPocS0[ StRpsIdx ][ i ]  
}  
NumPocStCurr0 = j  
  
for( i = 0, j = 0; i < NumPositivePics[ StRpsIdx ]; i++ ) {  
 if( UsedByCurrPicS1[ StRpsIdx ][ i ] )  
 PocStCurr1[ j++ ] = PicOrderCntVal + DeltaPocS1[ StRpsIdx ][ i ]  
 else  
 PocStFoll[ k++ ] = PicOrderCntVal + DeltaPocS1[ StRpsIdx ][ i ]  
}  
NumPocStCurr1 = j  
NumPocStFoll = k (8‑9)

for( i = 0, j = 0, k = 0; i < num\_long\_term\_pics; i++ ) {  
 if( delta\_poc\_msb\_present\_flag[ i ] = = 1 ) {  
 if( used\_by\_curr\_pic\_lt\_flag[ i ] )  
 PocLtCurr[ j++ ] = ( ( PicOrderCntVal − DeltaPocLt[ i ] + MaxPicOrderCntLsb ) %   
 MaxPicOrderCntLsb ) – ( DeltaPocMSBCycleLt[ i ] )\* MaxPicOrderCntLsb  
 else  
 PocLtFoll[ k++ ] = ( ( PicOrderCntVal − DeltaPocLt[ i ] + MaxPicOrderCntLsb ) %  
 MaxPicOrderCntLsb ) – ( DeltaPocMSBCycleLt[ i ] ) \* MaxPicOrderCntLsb  
 }  
 else{  
 if( used\_by\_curr\_pic\_lt\_flag[ i ] )  
 PocLtCurr[ j++ ] = ( PicOrderCntVal − DeltaPocLt[ i ] + MaxPicOrderCntLsb ) %   
 MaxPicOrderCntLsb  
 else  
 PocLtFoll[ k++ ] = ( PicOrderCntVal − DeltaPocLt[ i ] + MaxPicOrderCntLsb ) %  
 MaxPicOrderCntLsb  
 }  
 }

NumPocLtCurr = j  
NumPocLtFoll = k

where PicOrderCntVal is the picture order count of the current picture as specified in subclause 8.2.1.

NOTE 2 – A value of StRpsIdx in the range from 0 to num\_short\_term\_ref\_pic\_sets – 1, inclusive, indicates that a short-term reference picture set from the active parameter set is being used, where StRpsIdx is the index of the short-term reference picture set to the list of short-term reference picture sets in the order in which they are signalled in the picture parameter set. StRpsIdx equal to num\_short\_term\_ref\_pic\_sets indicates that a short-term reference picture set explicitly signalled in the slice header is being used.

The reference picture set consists of five lists of reference pictures; RefPicSetStCurr0, RefPicSetStCurr1, RefPicSetStFoll, RefPicSetLtCurr and RefPicSetLtFoll.

NOTE 3 –RefPicSetStCurr0, RefPicSetStCurr1 and RefPicSetLtCurr contains all reference pictures that may be used in inter prediction of the current picture and that may be used in inter prediction of one or more of the pictures following the current picture in decoding order. RefPicSetStFoll and RefPicSetLtFoll consists of all reference pictures that are *not* used in inter prediction of the current picture but may be used in inter prediction of one or more of the pictures following the current picture in decoding order.

The marking of a reference picture can be "unused for reference", "used for short-term reference", or "used for long-term reference", but only one among these three. When a reference picture is referred to as being marked as "used for reference", this collectively refers to the picture being marked as "used for short-term reference" or "used for long-term reference" (but not both). A reference picture that is marked as "used for short-term reference" is referred to as a short‑term reference picture. A reference picture that is marked as "used for long-term reference" is referred to as a long‑term reference picture*.*

Derivation process for the reference picture set and picture marking shall be performed according to the following ordered steps, where DPB refers to the decoded picture buffer as described in Annex C:

1. The following applies:

for( i = 0; i < NumPocLtCurr; i++ ) {

if(delta\_poc\_msb\_present\_flag[i] ==0){  
 if( there is a long-term reference picture picX in the DPB  
 with pic\_order\_cnt\_lsb equal to PocLtCurr[ i ] )  
 RefPicSetLtCurr[ i ] = picX  
 else if( there is a short-term reference picture picY in the DPB  
 with pic\_order\_cnt\_lsb equal to PocLtCurr[ i ] )  
 RefPicSetLtCurr[ i ] = picY  
 else   
 RefPicSetLtCurr[ i ] = "no reference picture"

}

else{

if( there is a long-term reference picture picX in the DPB  
 with PicOrderCntVal equal to PocLtCurr[ i ] )  
 RefPicSetLtCurr[ i ] = picX  
 else if( there is a short-term reference picture picY in the DPB  
 with PicOrderCntVal equal to PocLtCurr[ i ] )  
 RefPicSetLtCurr[ i ] = picY  
 else   
 RefPicSetLtCurr[ i ] = "no reference picture"   
}

}

for( i = 0; i < NumPocLtFoll; i++ ) {

if(delta\_poc\_msb\_present\_flag[i] ==0){  
 if( there is a long-term reference picture picX in the DPB  
 with pic\_order\_cnt\_lsb equal to PocLtFoll[ i ] )  
 RefPicSetLtFoll[ i ] = picX  
 else if( there is a short-term reference picture picY in the DPB  
 with pic\_order\_cnt\_lsb equal to PocLtFoll[ i ] )  
 RefPicSetLtFoll[ i ] = picY  
 else   
 RefPicSetLtFoll[ i ] = "no reference picture"

}

else{

if( there is a long-term reference picture picX in the DPB  
 with PicOrderCntVal to PocLtFoll[ i ] )  
 RefPicSetLtFoll[ i ] = picX  
 else if( there is a short-term reference picture picY in the DPB  
 with PicOrderCntVal equal to PocLtFoll[ i ] )  
 RefPicSetLtFoll[ i ] = picY  
 else   
 RefPicSetLtFoll[ i ] = "no reference picture"  
}

}

1. All reference pictures included in RefPicSetLtCurr and RefPicSetLtFoll are marked as "used for long-term reference"
2. The following applies:

for( i = 0; i < NumPocStCurr0; i++ ) {  
 if( there is a short-term reference picture picX in the DPB  
 with PicOrderCntVal equal to PocStCurr0[ i ])  
 RefPicSetStCurr0[ i ] = picX  
 else  
 RefPicSetStCurr0[ i ] = "no reference picture"  
}

for( i = 0; i < NumPocStCurr1; i++ ) {  
 if( there is a short-term reference picture picX in the DPB  
 with PicOrderCntVal equal to PocStCurr1[ i ])  
 RefPicSetStCurr1[ i ] = picX  
 else  
 RefPicSetStCurr1[ i ] = "no reference picture"  
} (8‑9)

for( i = 0; i < NumPocStFoll; i++ ) {  
 if( there is a short-term reference picture picX in the DPB  
 with PicOrderCntVal equal to PocStFoll[ i ])  
 RefPicSetStFoll[ i ] = picX  
 else  
 RefPicSetStFoll[ i ] = "no reference picture"  
}

1. All reference pictures included in RefPicSetStCurr0, RefPicSetStCurr1 and RefPicSetStFoll are marked as "used for short-term reference".
2. All reference pictures in the decoded picture buffer that are not included in RefPicSetLtCurr, RefPicSetLtFoll, RefPicSetStCurr0, RefPicSetStCurr1 or RefPicSetStFoll are marked as "unused for reference".

NOTE 4 – There may be one or more reference pictures that are included in the reference picture set but that are not present in the decoded picture buffer. Entries in RefPicSetStFoll or RefPicSetLtFoll equal to "no reference picture" should be ignored. An unintentional picture loss should be inferred for each entry in RefPicSetStCurr0, RefPicSetStCurr1 and RefPicSetLtCurr equal to "no reference picture".

The reference picture set is restricted as follows:

* There shall be no reference picture with temporal\_id greater than that of the current picture included in RefPicSetStCurr0, RefPicSetStCurr1 and RefPicSetLtCurr.
* There shall be no reference picture included in the reference picture set which precedes, in output order, any CRA picture that precedes the current picture both in decoding order and output order.
* There shall be no entry in RefPicSetStCurr0, RefPicSetStCurr1 or RefPicSetLtCurr equal to "no reference picture".

NOTE 5 – A reference picture can not be included in more than one of the five reference picture set lists.

### Decoding process for reference picture lists construction

This process is invoked at the beginning of the decoding process for each P, SP, or B slice.

Decoded reference pictures are marked as "used for short-term reference" or "used for long-term reference" as specified by the bitstream and specified in subclause 8.2.2.4. Short-term reference pictures are identified by the value of frame\_num. Long-term reference pictures are assigned a long-term picture index as specified by the bitstream and specified in subclause 8.2.2.4.

Subclause 8.2.2.1 is invoked to specify

– the assignment of variables FrameNum, FrameNumWrap, and PicNum to each of the short-term reference pictures, and

– the assignment of variable LongTermPicNum to each of the long-term reference pictures.

Reference pictures are addressed through reference indices as specified in subclause **Error! Reference source not found.**. A reference index is an index into a reference picture list. When decoding a P or SP slice, there is a single reference picture list RefPicList0. When decoding a B slice, there is a second independent reference picture list RefPicList1 in addition to RefPicList0.

At the beginning of the decoding process for each slice, reference picture list RefPicList0, and for B slices RefPicList1, are derived as specified by the following ordered steps:

1. An initial reference picture list RefPicList0 and for B slices RefPicList1 are derived as specified in subclause 8.2.2.2.
2. When ref\_pic\_list\_modification\_flag\_l0 is equal to 1 or, when decoding a B slice, ref\_pic\_list\_modification\_flag\_l1 is equal to 1, the initial reference picture list RefPicList0 and, for B slices, RefPicList1 are modified as specified in subclause .

NOTE – The modification process for reference picture lists specified in subclause  allows the contents of RefPicList0 and for B slices RefPicList1 to be modified in a flexible fashion. In particular, it is possible for a picture that is currently marked "used for reference" to be inserted into RefPicList0 and for B slices RefPicList1 even when the picture is not in the initial reference picture list derived as specified in subclause 8.2.2.2.

The number of entries in the modified reference picture list RefPicList0 is num\_ref\_idx\_l0\_active\_minus1 + 1, and for B slices the number of entries in the modified reference picture list RefPicList1 is num\_ref\_idx\_l1\_active\_minus1 + 1. A reference picture may appear at more than one index in the modified reference picture lists RefPicList0 or RefPicList1.

#### Initialisation process for reference picture lists

This initialisation process is invoked when decoding a P or B slice header.

When decoding a P or B slice, there shall be at least one reference picture in RefPicSetStCurr0, RefPicSetStCurr1 or RefPicSetLtCurr.

The following procedure is conducted to construct the initial RefPicList0:

cIdx = 0  
while( cIdx <= num\_ref\_idx\_l0\_active\_minus1 )  
{  
 for( i=0; i < NumPocStCurr0 && cIdx <= num\_ref\_idx\_l0\_active\_minus1; cIdx++, i++ )  
 RefPicList0[ cIdx ] = RefPicSetStCurr0[ i ]  
 for( i=0; i < NumPocStCurr1 && cIdx <= num\_ref\_idx\_l0\_active\_minus1; cIdx++, i++ ) (8‑9)  
 RefPicList0[ cIdx ] = RefPicSetStCurr1[ i ]  
 for( i=0; i < NumPocLtCurr && cIdx <= num\_ref\_idx\_l0\_active\_minus1; cIdx++, i++ )  
 RefPicList0[ cIdx ] = RefPicSetLtCurr[ i ]  
}

The following procedure is conducted to construct the initial RefPicList1:

cIdx = 0  
while( cIdx <= num\_ref\_idx\_l1\_active\_minus1 ) {  
 for( i=0; i < NumPocStCurr1 && cIdx <= num\_ref\_idx\_l1\_active\_minus1; cIdx++, i++ )  
 RefPicList1[ cIdx ] = RefPicSetStCurr1[ i ]   
 for( i=0; i < NumPocStCurr0 && cIdx <= num\_ref\_idx\_l1\_active\_minus1; cIdx++, i++ ) (8‑9)  
 RefPicList1[ cIdx ] = RefPicSetStCurr0[ i ]  
 for( i=0; i < NumPocLtCurr && cIdx <= num\_ref\_idx\_l1\_active\_minus1; cIdx++, i++ )  
 RefPicList1[ cIdx ] = RefPicSetLtCurr[ i ]  
}

#### Modification process for reference picture lists

After the invocation of this process, there shall be no reference pictures with greater temporal\_id than the current slice included in the output RefPicList0 or RefPicList1.

When ref\_pic\_list\_modification\_flag\_l0 is equal to 1, the following applies:

1. Let refIdxL0 be an index into the reference picture list RefPicList0. It is initially set equal to 0.
2. The corresponding syntax elements modification\_of\_pic\_nums\_idc are processed in the order they occur in the bitstream. For each of these syntax elements, the following applies.

– If modification\_of\_pic\_nums\_idc is equal to 0 or equal to 1, the process specified in subclause 8.2.2.3.1 is invoked with refIdxL0 as input, and the output is assigned to refIdxL0.

– Otherwise, if modification\_of\_pic\_nums\_idc is equal to 2, the process specified in subclause 8.2.2.3.2 is invoked with refIdxL0 as input, and the output is assigned to refIdxL0.

– Otherwise (modification\_of\_pic\_nums\_idc is equal to 3), the modification process for reference picture list RefPicList0 is finished.

When the current slice is a B slice and ref\_pic\_list\_modification\_flag\_l1 is equal to 1, the following applies:

1. Let refIdxL1 be an index into the reference picture list RefPicList1. It is initially set equal to 0.
2. The corresponding syntax elements modification\_of\_pic\_nums\_idc are processed in the order they occur in the bitstream. For each of these syntax elements, the following applies.

– If modification\_of\_pic\_nums\_idc is equal to 0 or equal to 1, the process specified in subclause  is invoked with refIdxL1 as input, and the output is assigned to refIdxL1.

– Otherwise, if modification\_of\_pic\_nums\_idc is equal to 2, the process specified in subclause  is invoked with refIdxL1 as input, and the output is assigned to refIdxL1.

– Otherwise (modification\_of\_pic\_nums\_idc is equal to 3), the modification process for reference picture list RefPicList1 is finished.

##### Modification process of reference picture lists for short-term reference pictures

Input to this process is an index refIdxLX (with X being 0 or 1).

Output of this process is an incremented index refIdxLX.

The variable picNumLXNoWrap is derived as follows.

If ref\_pic\_list\_modification\_idc is equal to 0, the following applies.

– If the current reference picture list is RefPicList0, curRefPicSet is set to RefPicSetStCurr0.  
– Otherwise (the current reference picture list is RefPicList1), curRefPicSet is set to RefPicSetStCurr1.

– Otherwise, if ref\_pic\_list\_modification\_idc is equal to 1, the following applies.

– If the current reference picture list is RefPicList0, curRefPicSet is set to RefPicSetStCurr1.  
– Otherwise (the current reference picture list is RefPicList1), curRefPicSet is set to RefPicSetStCurr0.

– Otherwise, if ref\_pic\_list\_modification\_idc is equal to 2, curRefPicSet is set to RefPicSetLtCurr.

The variable pocLX is derived as follows.

pocLX = curRefPicSet[ ref\_pic\_set\_idx ] (8‑9)

The following procedure is conducted to place the picture picR with PicOrderCnt( picR ) equal to pocLX into the index position refIdxLX, shift the position of any other remaining pictures to later in the list, and increment the value of refIdxLX.

for( cIdx = num\_ref\_idx\_lX\_active\_minus1 + 1; cIdx > refIdxLX; cIdx− − )  
 RefPicListX[ cIdx ] = RefPicListX[ cIdx − 1]  
RefPicListX[ refIdxLX++ ] = pocLX  
nIdx = refIdxLX (8‑9)  
for( cIdx = refIdxLX; cIdx <= num\_ref\_idx\_lX\_active\_minus1 + 1; cIdx++ )   
 if( PicOrderCnt( RefPicListX[ cIdx ] ) != pocLX )  
 RefPicListX[ nIdx++ ] = RefPicListX[ cIdx ]

NOTE 2 – Within this pseudo-code procedure, the length of the list RefPicListX is temporarily made one element longer than the length needed for the final list. After the execution of this procedure, only elements 0 through num\_ref\_idx\_lX\_active\_minus1 of the list need to be retained.

#### Mapping process for reference picture lists combination in B slices

[Ed.: (WJ) needs to be checked once again. Try to find better way to represent]

This initialisation process is invoked when decoding a B slice header.

Input to this process are the reference picture list RefPicListX and num\_ref\_idx\_lX\_active\_minus1 with X being 0 or 1.

Outputs of this process are arrays PredLCToPredLx and RefIdxLCToRefIdxLx.

When the current slice is a B slice and ref\_pic\_list\_modification\_flag\_lc is equal to 0, the following ordered steps apply:

Let refIdxL0 and refIdxL1 be indices into the reference picture lists RefPicListL0 and RefPicListL1. They are initially set equal to 0.

Let refIdxLC be an index into PredLCToPredLx and RefIdxLCToRefIdxLx. It is initially set equal to 0.

The following process is repeated until refIdxL0 and refIdxL1 are both greater than num\_ref\_idx\_l0\_active\_minus1 and num\_ref\_idx\_l1\_active\_minus1, respectively:

– If refIdxL0 is less than or equal to num\_ref\_idx\_l0\_active\_minus1,

– If the entry RefPicListL0[ refIdxL0 ] is the first occurance of the reference picture,

PredLCToPredLx[ refIdxLC ] = Pred\_L0, (8‑9)  
 RefIdxLCToRefIdxLx[ refIdxLC++ ] = refIdxL0.

– refIdxL0++.

– If refIdxL1 is less than or equal to num\_ref\_idx\_l1\_active\_minus1 and ref\_pic\_list\_combination\_flag equal to 1,

– If the entry RefPicListL1[ refIdxL1 ] is the first occurance of the reference picture,

PredLCToPredLx[ refIdxLC ] = Pred\_L1, (8‑10) RefIdxLCToRefIdxLx[ refIdxLC++ ] = refIdxL1.

– refIdxL1++.

When the current slice is a B slice and ref\_pic\_list\_modification\_flag\_lc is equal to 1, the following ordered steps apply:

Let refIdxLC be an index into the reference picture list PredLCToPredLx and RefIdxLCToRefIdxLx. It is initially set equal to 0.

The corresponding syntax elements pic\_from\_list\_0\_flag and ref\_idx\_list\_curr are processed in the order they occur in the bitstream. For each of these syntax elements pairs, the following applies.

– If pic\_from\_list\_0\_flag is equal to 1,

PredLCToPredLx[ refIdxLC ] = Pred\_L0, (8‑11)

– Otherwise,

PredLCToPredLx[ refIdxLC ] = Pred\_L1 (8‑12)

– RefIdxLCToRefIdxLx[ refIdxLC++ ] = ref\_idx\_list\_curr

When refIdxLC is greater than num\_com\_ref\_list\_active\_minus1+ 1, the extra entries past position num\_com\_ref\_list\_active\_minus1 are discarded from PredLCToPredLx and RefIdxLCToRefIdxLx.

When refIdxLC is less than num\_com\_ref\_list\_active\_minus1 + 1, the remaining entries in PredLCToPredLx and RefIdxLCToRefIdxLx are set equal to Pred\_L0 and 0, respectively.

### Marking of reference pictures before decoding

This process is invoked once per picture, after decoding of a slice header and the decoding process for reference picture set as specified in 0.1.3 but prior to the decoding of any coding unit and prior to the decoding process for reference picture list construction of the slice as specified in subclause 0.1.4.

When enable\_temporal\_mvp\_flag of the current picture is equal to 0 and temporal\_id of the current picture is equal to 0, all the reference pictures in the DPB are marked as “unused for temporal motion vector prediction”.

### Marking of the current picture after decoding

This process is invoked after all slices of the current picture have been decoded.

– If nal\_ref\_idc of the current picture equals 0, the current picture is marked as "unused for reference".

* Otherwise, the current picture is marked as "used for short-term reference".
* ---