Draft Text Modification

Added parts are shown in yellow, deleted parts are shown in green.

* + - * 1. Video parameter set extension 2 syntax

|  |  |
| --- | --- |
| vps\_extension2( ) { | **Descriptor** |
| while( !byte\_aligned( ) ) |  |
| **vps\_extension\_byte\_alignment\_reserved\_one\_bit** | u(1) |
| for( i = 0; i <= vps\_max\_layers\_minus1; i++ ) { |  |
| layerId = layer\_id\_in\_nuh[ i ] |  |
| if ( layerId ! = 0 ) { |  |
| **cp\_present\_flag**[ layerId ] | u(1) |
| if ( !VpsDepthFlag[ layerId ] ) { |  |
| **iv\_mv\_pred\_flag**[ layerId ] | u(1) |
| **iv\_res\_pred\_flag**[ layerId ] | u(1) |
| **log2\_sub\_PU\_size\_minus2**[layerId ] | ue(v) |
| **depth\_refinement\_flag**[ layerId ] | u(1) |
| **view\_synthesis\_pred\_flag**[ layerId ] | u(1) |
| } else { |  |
| **vps\_depth\_modes\_flag**[ layerId ] | u(1) |
| **lim\_qt\_pred\_flag**[ layerId ] | u(1) |
| if( vps\_depth\_modes\_flag[ layerId ] ) |  |
| **dlt\_flag**[ layerId ] | u(1) |
| if( dlt\_flag[ layerId ] ) { |  |
| **num\_depth\_values\_in\_dlt**[ layerId ] | ue(v) |
| for ( j = 0; j < num\_depth\_values\_in\_dlt[ layerId ] ; j++) { |  |
| **dlt\_depth\_value**[ layerId ][ j ] | ue(v) |
| } |  |
| } |  |
| } |  |
| } |  |
| } |  |
| **cp\_precision** | ue(v) |
| **cp\_in\_slice\_segment\_header\_flag** | u(1) |
| if( !cp\_in\_slice\_segment\_header\_flag ) |  |
| for( i = 0; i <= vps\_max\_layers\_minus1; i++ ) |  |
| layerId = layer\_id\_in\_nuh[ i ] |  |
| if( cp\_present\_flag[ layerId ] ) { |  |
| viewIdx = ViewOrderIdx[ layerId ] |  |
| for( j = 0; j < viewIdx; j++ ) { |  |
| **vps\_cp\_scale**[ viewIdx ][ j ] | se(v) |
| **vps\_cp\_off**[ viewIdx ][ j ] | se(v) |
| **vps\_cp\_inv\_scale\_plus\_scale**[ viewIdx ][ j ] | se(v) |
| **vps\_cp\_inv\_off\_plus\_off**[ viewIdx ][ j ] | se(v) |
| } |  |
| **}** |  |
| **iv\_mv\_scaling\_flag** | u(1) |
| } |  |

* + - * 1. Video parameter set extension 2 semantics

**log2\_sub\_PU\_size\_minus2** [layerId] specifies the value of the variable SubPUSize[layerId] that is used in the decoding process for inter-view motion prediction as follows:

SubPUSize[layerId] = 2( log2\_sub\_PU\_size\_minus2[layerId]+ 2 )

The value of log2\_sub\_PU\_size\_minus2 shall be in the range of 0 to 4, inclusive.

* + - 1. General

The specifications in subclause 8.5.3.1 apply with the following modification:

– All invocations of the process specified in subclause 8.5.3.2 are replaced with invocations of the process specified in subclause .

– All invocations of the process specified in subclause 8.5.3.3 are replaced with invocations of the process specified in subclause .

Inputs to this process are:

– a luma location ( xC, yC ) specifying the top-left sample of the current luma coding block relative to the top left luma sample of the current picture,

– a luma location ( xB, yB ) specifying the top-left sample of the current luma prediction block relative to the top left sample of the current luma coding block,

– a variable nCS specifying the size of the current luma coding block,

– a variable nPbW specifying the width of the current luma prediction block,

– a variable nPbH specifying the width of the current luma prediction block,

* a variable partIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are:

– a (nCSL)x(nCSL) array predSamplesL of luma prediction samples, where nCSL is derived as specified below,

– a (nCSC)x(nCSC) array predSamplesCb of chroma prediction samples for the component Cb, where nCSC is derived as specified below,

– a (nCSC)x(nCSC) array predSamplesCr of chroma prediction samples for the component Cr, where nCSC is derived as specified below.

The variable nCSL is set equal to nCS and the variable nCSC is set equal to nCS >> 1.

The decoding process for prediction units in inter prediction mode consists of the following ordered steps:

1. The derivation process for motion vector components and reference indices as specified in subclause H. is invoked with the luma coding block location ( xC, yC ), the luma prediction block location ( xB, yB ), the luma coding block size block nCS, the luma prediction block width and height, nPbW and nPbH, and the prediction unit index partIdx specifying as inputs and the luma motion vectors mvL0 and mvL1, the chroma motion vectors mvCL0 and mvCL1, the reference indices refIdxL0 and refIdxL1, and the prediction list utilization flags predFlagL0 and predFlagL1, the luma motion vector arrays mvL0IvMC and mvL1IvMC, the chroma motion vector arrays mvCL0IvMC and mvCL1IvMC, the reference index arrays refIdxL0IvMC and refIdxL1IvMC, and the prediction list utilization flag arrays predFlagL0IvMC and predFlagL1IvMC, and the flag ivMCFlag as outputs.
2. When the ivMCFlag is equal to 0, the following applies:
   1. The decoding process for inter sample prediction as specified in subclause H. is invoked with the luma coding block location ( xC, yC ), the luma prediction block location ( xB, yB ), the luma coding block size block nCS, the luma prediction block width height, nPbW and nPbH, the luma motion vectors mvL0 and mvL1, the chroma motion vectors mvCL0 and mvCL1, the reference indices refIdxL0 and refIdxL1, and the prediction list utilization flags predFlagL0 and predFlagL1 as inputs and the inter prediction samples (predSamples) which are a (nCSL)x(nCSL) array predSamplesL of prediction luma samples and two (nCSC)x(nCSC) arrays predSamplesCr, and predSamplesCr of prediction chroma samples, one for each of the chroma components Cb and Cr as outputs.
   2. For use in derivation processes of variables invoked later in the decoding process, the following assignments are made for x = xB..xB + nPbW − 1, y = yB..yB + nPbH− 1:

MvL0[ x ][ y ] = mvL0 (8‑61)  
MvL1[ x ][ y ] = mvL1 (8‑62)

RefIdxL0[ x ][ y ] = refIdxL0 (8‑63)  
RefIdxL1[ x ][ y ] = refIdxL1 (8‑64)

PredFlagL0[ x ][ y ] = predFlagL0 (8‑65)  
PredFlagL1[ x ][ y ] = predFlagL1 (8‑66)

1. When the ivMCFlag is equal to 1, the variables nSPW and nSPH are derived as

nSPW = nPSW /SubPUSize[nuh\_layer\_id]<=1 ? nPSW: SubPUSize[nuh\_layer\_id]

nSPH = nPSH /SubPUSize[nuh\_layer\_id]<=1 ? nPSH: SubPUSize[nuh\_layer\_id]

For each block (blkX=0..nPbW/ nSPW -1, blkY=0..nPbH/ nSPH-1), the following applies:

* 1. The decoding process for inter sample prediction as specified in subclause H. is invoked with the luma coding block location ( xC, yC ), the luma prediction block location ( xB+blkX\* nSPW, yB+blkY\* nSPH ), the luma coding block size block width nSPW, the luma prediction block height nSPH, the luma motion vectors mvL0IvMC[blkX][blkY] and mvL1IvMC[blkX][blkY], the chroma motion vectors mvCL0IvMC[blkX][blkY] and mvCL1IvMC[blkX][blkY], the reference indices refIdxL0IvMC[blkX][blkY] and refIdxL1IvMC[blkX][blkY], and the prediction list utilization flags predFlagL0IvMC[blkX][blkY] and predFlagL1IvMC[blkX][blkY] as inputs and the inter prediction samples (predSamples) which are a nSPWxnSPH array spPredSamplesL of prediction luma samples and two nSPW/2xnSPH/2 arrays spPredSamplesCr, and spPredSamplesCr of prediction chroma samples, one for each of the chroma components Cb and Cr as outputs
  2. For each luma sample (x=0.. nSPW-1, y=0.. nSPH-1), the following applies:

PredSamplesL[blkX\*nSPW+x][blkY\*nSPH+y] = spPredSamplesL[x][y]

* 1. For each chroma sample (x=0.. nSPW/2-1, y=0.. nSPH/2-1), the following applies:

PredSamplesCb[blkX\*(nSPW/2)+x][blkY\*(nSPH/2)+y] = spPredSamplesCb[x][y]

PredSamplesCr[blkX\*(nSPW/2)+x][blkY\*(nSPH/2)+y] = spPredSamplesCr[x][y]

* 1. For use in derivation processes of variables invoked later in the decoding process, the following assignments are made for x=xB..xB+ nSPW-1, y=yB..yB+ nSPH-1:

MvL0[ blkX\* nSPW +x ][ blkY\* nSPH+y ] = mvL0IvMC[blkX][blkY] (8‑61)  
MvL1[ blkX\* nSPW +x ][ blkY\* nSPH+y ] = mvL1IvMC[blkX][blkY] (8‑62)

RefIdxL0[ blkX\* nSPW +x ][ blkY\* nSPH +y ] = refIdxL0IvMC[blkX][blkY] (8‑63)  
RefIdxL1[ blkX\* nSPW +x ][ blkY\* nSPH +y ] = refIdxL1IvMC[blkX][blkY] (8‑64)

PredFlagL0[ blkX\* nSPW+x ][ blkY\* nSPH+y ] = predFlagL0IvMC[blkX][blkY] (8‑65)  
PredFlagL1[ blkX\* nSPW+x ][ blkY\* nSPH+y ] = predFlagL1IvMC[blkX][blkY] (8‑66)

* + - 1. Derivation process for motion vector components and reference indices

Inputs to this process are:

* a luma location ( xCb, yCb ) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
* a luma location ( xBl, yBl ) of the top-left sample of the current luma prediction block relative to the top-left sample of the current luma coding block,
* a variable nCbS specifying the size of the current luma coding block,
* two variables nPbW and nPbH specifying the width and the height of the luma prediction block,
* a variable partIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are:

* the luma motion vectors mvL0 and mvL1,
* the chroma motion vectors mvCL0 and mvCL1,
* the reference indices refIdxL0 and refIdxL1,
* the prediction list utilization flags predFlagL0 and predFlagL1.
* the luma motion vector arrays mvL0IvMC and mvL1IvMC,
* the chroma motion vector arrays mvCL0IvMC and mvCL1IvMC,
* the reference index arrays refIdxL0IvMC and refIdxL1IvMC,
* the prediction list utilization flag arrays predFlagL0IvMC and predFlagL1IvMC,
* and the flag ivMCFlag

Let ( xPb, yPb ) specify the top-left sample location of the current luma prediction block relative to the top-left luma sample of the current picture where xPb = xCb + xBl and yPb = yCb + yBl.

Let the variable currPic and ListX be the current picture and RefPicListX, with X being 0 or 1, of the current picture, respectively.

The function LongTermRefPic( aPic, aPb, refIdx, LX ), with X being 0 or 1, is defined as follows:

* If the picture with index refIdx from reference picture list LX of the slice containing prediction block aPb in the picture aPic was marked as "used for long term reference" at the time when aPic was the current picture, LongTermRefPic( aPic, aPb, refIdx, LX ) is equal to 1.
* Otherwise, LongTermRefPic( aPic, aPb, refIdx, LX ) is equal to 0.

The variables vspModeFlag, ivpMvFlagL0 and ivpMvFlagL1 are set equal to 0.

For the derivation of the variables mvL0 and mvL1, refIdxL0 and refIdxL1, as well as predFlagL0 and predFlagL1, the following applies:

* If merge\_flag[ xPb ][ yPb ] is equal to 1, the derivation process for luma motion vectors for merge mode as specified in subclause is invoked with the luma location ( xCb, yCb ), the luma location ( xPb, yPb ), the variables nCbS, nPbW, nPbH, and the partition index partIdx as inputs, and the output being the luma motion vectors mvL0, mvL1, the reference indices refIdxL0, refIdxL1, and the prediction list utilization flags predFlagL0 and predFlagL1,, the disparity vector availability flags ivpMvFlagL0 and ivpMvFlagL1, the flag vspModeFlag, the luma motion vector arrays mvL0IvMC and mvL1IvMC, the reference index arrays refIdxL0IvMC and refIdxL1IvMC, the prediction list utilization flag arrays predFlagL0IvMC and predFlagL1IvMC, and the flag ivMCFlag .
* Otherwise, for X being replaced by either 0 or 1 in the variables predFlagLX, mvLX, and refIdxLX, in PRED\_LX, and in the syntax elements ref\_idx\_lX and MvdLX, the following applies:
  + 1. The variables refIdxLX and predFlagLX are derived as follows:
       - * If inter\_pred\_idc[ xPb ][ yPb ] is equal to PRED\_LX or PRED\_BI,

refIdxLX = ref\_idx\_lX[ xPb ][ yPb ] (‑69)

predFlagLX = 1 (‑70)

* + - * + Otherwise, the variables refIdxLX and predFlagLX are specified by:

refIdxLX = −1 (‑71)

predFlagLX = 0 (‑72)

* + 1. The variable mvdLX is derived as follows:

mvdLX[ 0 ] = MvdLX[ xPb ][ yPb ][ 0 ] (‑73)

mvdLX[ 1 ] = MvdLX[ xPb ][ yPb ][ 1 ] (‑74)

* + 1. When predFlagLX is equal to 1, the derivation process for luma motion vector prediction in subclause 8.5.3.2.5 is invoked with the luma coding block location ( xCb, yCb ), the coding block size nCbS, the luma prediction block location ( xPb, yPb ), the variables nPbW, nPbH, refIdxLX, and the partition index partIdx as inputs, and the output being mvpLX.
    2. When predFlagLX is equal to 1, the luma motion vector mvLX is derived as follows:

uLX[ 0 ] = ( mvpLX[ 0 ] + mvdLX[ 0 ] + 216 ) % 216 (‑75)

mvLX[ 0 ] = ( uLX[ 0 ] >= 215 ) ? ( uLX[ 0 ] − 216 ) : uLX[ 0 ] (‑76)

uLX[ 1 ] = ( mvpLX[ 1 ] + mvdLX[ 1 ] + 216 ) % 216 (‑77)

mvLX[ 1 ] = ( uLX[ 1 ] >= 215 ) ? ( uLX[ 1 ] − 216 ) : uLX[ 1 ] (‑78)

NOTE – The resulting values of mvLX[ 0 ] and mvLX[ 1 ] as specified above will always be in the range of −215 to 215 − 1, inclusive.

When ChromaArrayType is not equal to 0 and predFlagLX, with X being 0 or 1, is equal to 1, the following applies:

* + - 1. when the flag ivMCFlag is equal to 0, the derivation process for chroma motion vectors in subclause 8.5.3.2.9 is invoked with mvLX as input, and the output being mvCLX.
      2. When the flag ivMCFlag is equal to 1, the variables nSPW and nSPH are derived as

nSPW = nPSW /SubPUSize[nuh\_layer\_id]<=1

? nPSW: SubPUSize[nuh\_layer\_id]

nSPH = nPSH /SubPUSize[nuh\_layer\_id]<=1

? nPSH: SubPUSize[nuh\_layer\_id]

for each block (blkX=0..nPbW/nSPW-1, blkY=0..nPbH/nSPH-1), the derivation process for chroma motion vectors in subclause 8.5.3.2.9 is invoked with mvLXIvMC[blkX][blkY] and refIdxLXIvMC[blkX][blkY] as inputs and the output being mvCLXIvMC[blkX][blkY].

For use in derivation processes of variables invoked later in the decoding process, the following assignments are made for x = xPb.. ( xPb + nPbW − 1 ), y = yPb..( yPb + nPbH− 1 ) (with X being either 0 or 1):

* 1. IvpMvFlagLX[ x ][ y ] = ivpMvFlagLX (‑79)  
     VspModeFlag[ x ][ y ] = vspModeFlag (‑80)
     + - 1. Derivation process for luma motion vectors for merge mode

This process is only invoked when merge\_flag[ xPb ][ yPb ] is equal to 1, where ( xPb, yPb ) specify the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture.

Inputs to this process are:

* a luma location ( xCb, yCb ) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
* a luma location ( xPb, yPb ) of the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,
* a variable nCbS specifying the size of the current luma coding block,
* two variables nPbW and nPbH specifying the width and the height of the luma prediction block,
* a variable partIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are:

* the luma motion vectors mvL0 and mvL1,
* the reference indices refIdxL0 and refIdxL1,
* the prediction list utilization flags predFlagL0 and predFlagL1,
* the disparity vector availability flags ivpMvFlagL0 and ivpMvFlagL1,
* the flag vspModeFlag, specifying, whether the current PU is coded using view synthesis prediction,
* the luma motion vector arrays mvL0IvMC and mvL1IvMC,
* the reference index arrays refIdxL0IvMC and refIdxL1IvMC,
* the prediction list utilization flag arrays predFlagL0IvMC and predFlagL1IvMC,
* the flag ivMCFlag

The location ( xOrigP, yOrigP ) and the variables nOrigPbW and nOrigPbH are derived to store the values of ( xPb, yPb ), nPbW, and nPbH as follows:

* + - 1. ( xOrigP, yOrigP ) is set equal to ( xPb, yPb ) (‑81)
      2. nOrigPbW = nPbW (‑82)
      3. nOrigPbH = nPbH (‑83)

When Log2ParMrgLevel is greater than 2 and nCbS is equal to 8, ( xPb, yPb ), nPbW, nPbH, and partIdx are modified as follows:

* + - 1. ( xPb, yPb ) = ( xCb, yCb ) (‑84)
      2. nPbW = nCbS (‑85)
      3. nPbH = nCbS (‑86)
      4. partIdx = 0 (‑87)

NOTE – When Log2ParMrgLevel is greater than 2 and nCbS is equal to 8, all the prediction units of the current coding unit share a single merge candidate list, which is identical to the merge candidate list of the 2Nx2N prediction unit.

The motion vectors mvL0 and mvL1, the reference indices refIdxL0 and refIdxL1, and the prediction utilization flags predFlagL0 and predFlagL1, the luma motion vector arrays mvL0IvMC and mvL1IvMC, the reference index arrays refIdxL0IvMC and refIdxL1IvMC, the prediction list utilization flag arrays predFlagL0IvMC and predFlagL1IvMC, the flag ivMCFlag are derived by the following ordered steps:

* 1. The derivation process for merging candidates from neighbouring prediction unit partitions in subclause 8.5.3.2.2 is invoked with the luma coding block location ( xCb, yCb ), the coding block size nCbS, the luma prediction block location ( xPb, yPb ), the luma prediction block width nPbW, the luma prediction block height nPbH, and the partition index partIdx as inputs, and the output being the availability flags availableFlagA0, availableFlagA1, availableFlagB0, availableFlagB1, and availableFlagB2, the reference indices refIdxLXA0, refIdxLXA1, refIdxLXB0, refIdxLXB1, and refIdxLXB2, the prediction list utilization flags predFlagLXA0, predFlagLXA1, predFlagLXB0, predFlagLXB1, and predFlagLXB2, and the motion vectors mvLXA0, mvLXA1, mvLXB0, mvLXB1, and mvLXB2, with X being 0 or 1.
  2. The reference indices for the temporal merging candidate, refIdxLXCol, with X being 0 or 1, are set equal to 0.
  3. The derivation process for temporal luma motion vector prediction in subclause is invoked with the luma location ( xPb, yPb ), the luma prediction block width nPbW, the luma prediction block height nPbH, and the variable refIdxL0Col as inputs, and the output being the availability flag availableFlagL0Col and the temporal motion vector mvL0Col.The variables availableFlagCol, predFlagL0Col and predFlagL1Col are derived as follows:
     + 1. availableFlagCol = availableFlagL0Col (‑88)
       2. predFlagL0Col = availableFlagL0Col (‑89)
       3. predFlagL1Col = 0 (‑90)
  4. When slice\_type is equal to B, the derivation process for temporal luma motion vector prediction in subclause is invoked with the luma location ( xPb, yPb ), the luma prediction block width nPbW, the luma prediction block height nPbH, and the variable refIdxL1Col as inputs, and the output being the availability flag availableFlagL1Col and the temporal motion vector mvL1Col. The variables availableFlagCol and predFlagL1Col are derived as follows:
     + 1. availableFlagCol = availableFlagL0Col  | |  availableFlagL1Col (‑91)
       2. predFlagL1Col = availableFlagL1Col (‑92)
  5. Depending on iv\_mv\_pred\_flag[ nuh\_layer\_id ], the following applies.
     + If iv\_mv\_pred\_flag[ nuh\_layer\_id ] is equal to 0, the flags availableFlagIvMC, availableIvMCShift and availableFlagIvDC are set equal to 0.
     + Otherwise (iv\_mv\_pred\_flag[ nuh\_layer\_id ] is equal to 1), the derivation process for the inter-view merge candidates as specified in subclause is invoked with the luma location ( xPb, yPb ), the variables nPbW and nPbH, as the inputs and the output is assigned to the availability flags availableFlagIvMC, availableIvMCShift and availableFlagIvDC, the reference indices refIdxLXIvMC, refIdxLXIvMCShift and refIdxLXIvDC, the prediction list utilization flags predFlagLXIvMC, predFlagLXivMCShift and predFlagLXIvDC, and the motion vectors mvLXIvMC, mvLXIvMCShift and mvLXIvDC, the luma motion vector arrays mvLXIvMC, the reference index arrays refIdxLXIvMC, the prediction list utilization flag arrays predFlagLXIvMC (with X being 0 or 1, respectively).
  6. Depending on view\_synthesis\_pred\_flag[ nuh\_layer\_id ], the following applies.
     + If view\_synthesis\_pred\_flag[ nuh\_layer\_id ] is equal to 0, the flag availableFlagVSP is set equal to 0.
     + Otherwise (view\_synthesis\_pred\_flag[ nuh\_layer\_id ] is equal to 1), the derivation process for a view synthesis prediction merge candidate as specified in subclause is invoked with the luma locations ( xCb, yCb ) as input and the outputs are the availability flag availableFlagVSP, the reference indices refIdxL0VSP and refIdxL1VSP, the prediction list utilization flags predFlagL0VSP and predFlagL1VSP, and the motion vectors mvL0VSP and mvL1VSP.
  7. Depending on DepthFlag, the following applies.
     + If DepthFlag is equal to 0, the variable availableFlagT is set equal to 0.
     + Otherwise ( DepthFlag is equal to 1), the derivation process for the texture merging candidate as specified in subclause is invoked with the luma location ( xPb, yPb ), the variables nPbW and nPbH as the inputs and the outputs are the flag availableFlagT, the prediction utilization flags predFlagL0T and predFlagL1T, the reference indices refIdxL0T and refIdxL1T, and the motion vectors mvL0T and mvL1T.
  8. The merge candidate lists mergeCandList and mergeCandIsVspFlag are constructed as specified by the following ordered steps:
  9. The variable numMergeCand is set equal to 0.
  10. When availableFlagT is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to T, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to 0 and the variable numMergeCand is increased by 1.
  11. When availableFlagIvMC is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to IvMC, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to 0 and the variable numMergeCand is increased by 1.
  12. When availableFlagA1 is equal to 1, the following applies:
      + - When the following condition is true,
          * availableFlagT = = 0 && availableFlagIvMC  = = 0,
          1. or one or more of the following conditions are true, with N being replaced by T and IvMC:
          * availableFlagN = = 1 && predFlagLXN  !=  predFlagLXA1, (with X being replaced by 0 and 1),
          * availableFlagN = = 1 && mvLXN  !=  mvLXA1 (with X being replaced by 0 and 1),
          * availableFlagN = = 1 && refIdxLXN  !=  refIdxLXA1 (with X being replaced by 0 and 1),

the entry mergeCandList[ numMergeCand ] is set equal to A1, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to VspModeFlag[ xPb − 1 ][ yPb + nPbH − 1 ] and the variable numMergeCand is increased by 1.

* 1. When availableFlagB1 is equal to 1, the following applies:
     + - When the following condition is true ,
         * availableFlagT = = 0 && availableFlagIvMC = = 0,

or one or more of the following conditions is true, with N being replaced by T and IvMC:

* + - * + availableFlagN = = 1 && predFlagLXN  !=  predFlagLXB1, (with X being replaced by 0 and 1),
        + availableFlagN = = 1 && mvLXN  !=  mvLXB1 (with X being replaced by 0 and 1),
        + availableFlagN = = 1 && refIdxLXN  !=  refIdxLXB1 (with X being replaced by 0 and 1),

the entry mergeCandList[ numMergeCand ] is set equal to B1, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to VspModeFlag[ xPb + nPbW − 1 ][ yPb − 1 ] and the variable numMergeCand is increased by 1.

* 1. When availableFlagB0 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to B0, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to VspModeFlag[ xPb + nPbW ][ yPb − 1 ] and the variable numMergeCand is increased by 1.
  2. When availableFlagIvDC is equal to 1, and one or more of the following conditions is true,
     + - availableFlagA1  = =  0,
       - predFlagLXA1  !=  predFlagLXIvDC, (with X being replaced by 0 and 1),
       - mvLXA1  !=  mvLXIvDC(with X being replaced by 0 and 1),
       - refIdxLXA1  !=  refIdxLXIvDC(with X being replaced by 0 and 1),

and one or more of the following conditions is true,

* + - * availableFlagB1  = =  0,
      * predFlagLXB1  !=  predFlagLXIvDC, (with X being replaced by 0 and 1),
      * mvLXB1  !=  mvLXIvDC(with X being replaced by 0 and 1),
      * refIdxLXB1  !=  refIdxLXIvDC(with X being replaced by 0 and 1),

the entry mergeCandList[ numMergeCand ] is set equal to IvDC, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to 0 and the variable numMergeCand is increased by 1.

* 1. When availableFlagVSP is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to VSP, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal 1 and the variable numMergeCand is increased by 1.
  2. When availableFlagA0 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to A0, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to VspModeFlag[ xPb − 1 ][ yPb + nPbH ] and the variable numMergeCand is increased by 1.
  3. When availableFlagB2 is equal to 1 and numMergeCand is less than 4 + iv\_mv\_pred\_flag[ nuh\_layer\_id ] + DepthFlag, the entry mergeCandList[ numMergeCand ] is set equal to B2, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to VspModeFlag[ xPb − 1 ][ yPb − 1 ] and the variable numMergeCand is increased by 1.
  4. When availableFlagIvMCShift is equal to 1 and numMergeCand is less than 6, and one or more of the following conditions are true,
     + - availableFlagIvMC  = =  0,
       - predFlagLXMC  !=  predFlagLXMCShift(with X being replaced by 0 and 1),
       - mvLXMC  !=  mvLXIvMCShift(with X being replaced by 0 and 1),
       - refIdxLXMC  !=  refIdxLXMCShift(with X being replaced by 0 and 1),

the entry mergeCandList[ numMergeCand ] is set equal to IvMCShift, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to 0 and the variable numMergeCand is increased by 1.

* 1. A variable availableFlagIvDCShift is set to 0 and when all of the following conditions are true
     + - DepthFlag is equal to 0, [Ed (CY): this condition is already assumed thus could be removed.]
       - availableFlagIvMCShift is equal to 0,
       - numMergeCand is less than 6,

the derivation process for the shifted disparity merging candidate as specified in subclause is invoked with the availability flags availableFlagN, the reference indices refIdxL0N and refIdxL1N, the prediction list utilization flags predFlagL0N and predFlagL1N, the motion vectors mvL0N and mvL1N, of every candidate N being in mergeCandList, mergeCandList, mergeCandIsVspFlag, and numMergeCand as the inputs and the outputs are the flag availableFlagIvDCShift, the prediction utilization flags predFlagL0IvDCShift and predFlagL1IvDCShift, the reference indices refIdxL0IvDCShift and refIdxL1IvDCShift, and the motion vectors mvL0IvDCShift and mvL1IvDCShift. When availableFlagIvDCShift is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to IvDCShift, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to 0 and the variable numMergeCand is increased by 1.

* 1. When availableFlagCol is equal to 1 and numMergeCand is less than 5 + iv\_mv\_pred\_flag[ nuh\_layer\_id ] + DepthFlag, the entry mergeCandList[ numMergeCand ] is set equal to Col, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to 0 and the variable numMergeCand is increased by 1.
  2. The variable numOrigMergeCand is set equal to numMergeCand.
  3. When slice\_type is equal to B, the derivation process for combined bi-predictive merging candidates specified in subclause is invoked with mergeCandList, mergeCandIsVspFlag, the reference indices refIdxL0N and refIdxL1N, the prediction list utilization flags predFlagL0N and predFlagL1N, the motion vectors mvL0N and mvL1N of every candidate N in mergeCandList, numCurrMergeCand, and numOrigMergeCand as inputs, and the output is assigned to mergeCandList, numCurrMergeCand, the reference indices refIdxL0combCandk and refIdxL1combCandk, the prediction list utilization flags predFlagL0combCandk and predFlagL1combCandk, and the motion vectors mvL0combCandk and mvL1combCandk of every new candidate combCandk being added into mergeCandList. The number of candidates being added, numCombMergeCand, is set equal to ( numCurrMergeCand − numOrigMergeCand ). When numCombMergeCand is greater than 0, k ranges from 0 to numCombMergeCand − 1, inclusive, and mergeCandIsVspFlag[ numOrigMergeCand + k ] is set equal to 0.
  4. The derivation process for zero motion vector merging candidates specified in subclause 8.5.3.2.4 is invoked with the mergeCandList, the reference indices refIdxL0N and refIdxL1N, the prediction list utilization flags predFlagL0N and predFlagL1N, the motion vectors mvL0N and mvL1N of every candidate N in mergeCandList, and numCurrMergeCand as inputs, and the output is assigned to mergeCandList, numCurrMergeCand, the reference indices refIdxL0zeroCandm and refIdxL1zeroCandm, the prediction list utilization flags predFlagL0zeroCandm and predFlagL1zeroCandm, and the motion vectors mvL0zeroCandm and mvL1zeroCandm of every new candidate zeroCandm being added into mergeCandList. The number of candidates being added, numZeroMergeCand, is set equal to ( numCurrMergeCand − numOrigMergeCand − numCombMergeCand ). When numZeroMergeCand is greater than 0, m ranges from 0 to numZeroMergeCand − 1, inclusive, and mergeCandIsVspFlag[ numOrigMergeCand + numCombMergeCand + m ] is set equal to 0.
  5. The following assignments are made with N being the candidate at position merge\_idx[ xOrigP ][ yOrigP ] in the merging candidate list mergeCandList ( N = mergeCandList[ merge\_idx[ xOrigP ][ yOrigP ] ] ) and X being replaced by 0 or 1:

If N is not equal to IvMC, the following applies:

* + - 1. mvLX[ 0 ] = mvLXN[ 0 ] (‑93)
      2. mvLX[ 1 ] = mvLXN[ 1 ] (‑94)
      3. refIdxLX = refIdxLXN (‑95)
      4. predFlagLX = predFlagLXN (‑96)
      5. ivMCFlag = 0
    1. otherwise,
       1. ivMCFlag = 1
  1. When predFlagL0 is equal to 1 and predFlagL1 is equal to 1, and ( nOrigPbW + nOrigPbH ) is equal to 12, the following applies
     + 1. refIdxL1 = −1 (‑97)
       2. predFlagL1 = 0 (‑98)
  2. The variable vspModeFlag is set equal to mergeCandIsVspFlag[ merge\_idx[ xPb][ yPb ] ]
  3. When vspModeFlag is equal to 1, the following applies:
     + The variables xVsp and yVsp are derived depending on the value of mergeCandList[ merge\_idx[ xPb][ yPb ] ] as specified in .
     + The variable vspRefViewIdx is set equal RefViewIdx[ xVsp ][ yVsp ] and the variable vspMvDisp is set equal to MvDisp[ xVsp ][ yVsp ].
     + For x in the range of xPb to ( xPb + nPbW − 1 ), inclusive, the following applies:
       - For y in the range of yPb to ( yPb + nPbH − 1 ), inclusive, the following applies:
         1. RefViewIdx[ x ][ y ] = vspRefViewIdx (‑99)
         2. MvDisp[ x ][ y ] = vspMvDisp (‑100)
  4. The disparity availability flag ivpMvFlagLX is derived as follows (with X being replace by 0 or 1).
     + If one of the following conditions are true, ivpMvFlagLX is set equal to 1

[Ed. (GT) There is some redundancy in draft and software since disparities equal for both lists.(#7) ]

* + - * predFlagLXIvMC  = =  1 && mergeCandList[ merge\_idx[ xPb][ yPb ] ]  =  =  IvMC
      * predFlagLXIvMCShift  = =  1 && and mergeCandList[ merge\_idx[ xPb][ yPb ] ]  = =  IvMCShift

[Ed. (GT): PredMode[ xCb ][ yCb ]  = =  MODE\_SKIP might be added here instead of testing it in the disparity vector derivation process]

* + - Otherwise, ivpMvFlagLX is set equal to 0.
      * 2. Derivation process for inter-view merge candidates

This process is not invoked when iv\_mv\_pred\_flag[ nuh\_layer\_id ] is equal to 0.

Inputs to this process are:

* a luma location ( xPb, yPb ) of the top-left luma sample of the current prediction unit relative to the top-left luma sample of the current picture,
* variables nPbW and nPbH specifying the width and the height, respectively, of the current prediction unit,

Outputs of this process are (with X being 0 or 1, respectively)

* the availability flags availableFlagIvMC, availableFlagIvMCShift and availableFlagIvDC specifying whether the inter-view merge candidates are available,
* the reference indices refIdxLXIvMC, refIdxLXIvMCShift and refIdxLXIvDC,
* the prediction list utilization flags predFlagLXIvMC, predFlagLXIvMCShift and predFlagLXIvDC,
* the motion vectors mvLXIvMC, mvLXIvMCShift and mvLXIvDC,
* the luma motion vector arrays mvL0IvMC and mvL1IvMC,
* the reference index arrays refIdxL0IvMC and refIdxL1IvMC,
* the prediction list utilization flag arrays predFlagL0IvMC and predFlagL1IvMC

The temporal inter-view motion vector merging candidate is derived by the following ordered steps.

* + 1. For the prediction list indication X being 0 and 1 the following applies;
    - The derivation process for a temporal inter-view motion vector candidate as specified in subclause  is invoked with the luma location ( xPb, yPb ), the variables nPbW and nPbH, the prediction list indication X, the view order index RefViewIdx[ xPb ][ yPb ] and the disparity vector MvRefinedDisp[ xPb ][ yPb ] as the inputs and the outputs are the flag availableFlagLXIvMC, the motion vector mvLXIvMC and the reference index refIdxLXIvMC.
  1. The derivation process for a temporal inter-view motion vector candidate as specified in subclause 6 is invoked with the luma location ( xPb, yPb ), the variables nPbW and nPbH, the view order index RefViewIdx[ xPb ][ yPb ] and the disparity vector MvRefinedDisp[ xPb ][ yPb ] as the inputs and the outputs are the flag availableFlagLXIvMC, the motion vector mvLXIvMC and the reference index refIdxLXIvMC, the luma motion vector arrays mvLXIvMC, the reference index arrays refIdxLXIvMC, the prediction list utilization flag arrays predFlagLXIvMC (with X being 0 or 1).
  2. The availability flag availableFlagIvMC, and the prediction utilization flags predFlagL0IvMC and predFlagL1IvMC are derived by
     + 1. availableFlagIvMC = availableFlagL0IvMC | | availableFlagL1IvMC (‑)
       2. predFlagL0IvMC = availableFlagL0IvMC (‑)
       3. predFlagL1IvMC = availableFlagL1IvMC (‑)

The shifted temporal inter-view motion vector merging candidate is derived by the following ordered steps.

* 1. For the prediction list indication X being 0 and 1 the following applies.
     + The derivation process for a temporal inter-view motion vector candidate as specified in subclause  is invoked with the luma location ( xPb, yPb ), the variables nPbW and nPbH, the prediction list indication X , the view order index RefViewIdx[ xPb ][ yPb ], the disparity vector MvRefinedDisp[ xPb ][ yPb ] + ( nPbW \*2 + 4, nPbH \*2 + 4 ), and the reference index refIdxLX being equal to −1, and as the inputs and the outputs are the flag availableFlagLXIvMCShift, the motion vector mvLXIvMCShift and the reference index refIdxLXIvMCShift.
  2. The availability flag availableFlagIvMCShift, and the prediction utilization flags predFlagL0IvMCShift and predFlagL1IvMCShift are derived by
     + 1. availableFlagIvMCShift = availableFlagL0IvMCShift | | availableFlagL1IvMCShift (‑)
       2. predFlagL0IvMCShift = availableFlagL0IvMCShift (‑)
       3. predFlagL1IvMCShift = availableFlagL1IvMCShift (‑)

The disparity inter-view motion vector merging candidate is derived by the following ordered steps.

* 1. For the prediction list indication X being 0 and 1 the following applies.
     + The derivation process for a disparity inter-view motion vector candidate as specified in subclause  is invoked with the luma location ( xPb, yPb ), the variables nPbW and nPbH, the view order index RefViewIdx[ xPb ][ yPb ], the disparity vector MvRefinedDisp[ xPb ][ yPb ], and the prediction list indication X, as the inputs and the outputs are the flag availableFlagLXIvDC, the motion vector mvLXIvDC, and the reference index refIdxLXIvDC.
  2. The availability flag availableFlagIvDC, and the prediction utilization flags predFlagL0IvDC and predFlagL1IvDC are derived by
     + 1. availableFlagIvDC = availableFlagL0IvDC | | availableFlagL1IvDC  (‑)
       2. predFlagL0IvDC = availableFlagL0IvDC (‑)
       3. predFlagL1IvDC = availableFlagL1IvDC (‑)
          1. Derivation process for a sub-PU temporal inter-view motion vector candidate

This process is not invoked when iv\_mv\_pred\_flag[ nuh\_layer\_id ] is equal to 0.

Inputs to this process are:

* a luma location ( xPb, yPb ) of the top-left luma sample of the current prediction unit relative to the top-left luma sample of the current picture,
* variables nPbW and nPbH specifying the width and the height, respectively, of the current prediction unit,
* a reference view index refViewIdx.
* a disparity vector mvDisp,

Outputs of this process are (with X being to 0 or 1 respectively):

* a flag availableFlagLXInterView specifying whether the temporal inter-view motion vector candidate is available,
* a temporal inter-view motion vector candidate mvLXInterView,
* a reference index refIdxLX specifying a reference picture in the reference picture list RefPicListLX,
* the inter-view motion vector arrays spMvLXInterView,
* the reference index arrays spRefIdxLX,
* the prediction list utilization flag arrays spPredFlagLX

The flag availableFlagLXInterView is set equal to 0, both components of mvLXInterView are set equal to 0.

The variables nSPW and nSPH are derived as

nSPW = nPSW /SubPUSize[nuh\_layer\_id]<=1 ? nPSW: SubPUSize[nuh\_layer\_id]

nSPH = nPSH /SubPUSize[nuh\_layer\_id]<=1 ? nPSH: SubPUSize[nuh\_layer\_id]

For each block location (blkX = 0..nPSW/ nSPW -1, blkY=0..nPSH/ nSPH -1), the following applies:

* The reference layer luma location ( xRef, yRef ) is derived by

xRef = Clip3( 0, PicWidthInSamplesL – 1, xPb + blkX\* nSPW +nSPW/2 + ( ( mvDisp[ 0 ] + 2 ) >> 2 ) ) (‑124)  
yRef = Clip3( 0, PicHeightInSamplesL – 1, yPb + blkY\* nSPH +nSPH/2 + ( ( mvDisp[ 1 ] + 2 ) >> 2 ) ) (‑125)

* The variable ivRefPic is set equal to the picture with ViewIdx equal to refViewIdx in the current access unit.
* The variable ivRefPb specifies the luma prediction block covering the location given by ( xRef, yRef ) inside the inter-view reference picture specified by ivRefPic.
* The luma location ( xIvRefPb, yIvRefPb ) is set equal to the top-left sample of the inter-view reference luma prediction block specified by ivRefPb relative to the top-left luma sample of the inter-view reference picture specified by ivRefPic.
* When ivRefPb is not coded in an intra prediction mode, the following applies, for X equal to 0 and 1, Y in the range of X to (1 – X), inclusive:
  + The spRefIdxLX[blkX][blkY] is set equal to -1. The flag availableFlagLXInterView is set to 0. The flag spPredFlagLX[blkX][blkY] is set to 0.
  + When X is equal to 1 and the current slice is not a B slice the whole decoding process specified in this subclause terminates.
  + The variables refPicListLYIvRef, predFlagLYIvRef[ x ][ y ], mvLYIvRef[ x ][ y ], and refIdxLYIvRef[ x ][ y ] are set equal to the corresponding variables of the inter-view reference picture specified by ivRefPic, RefPicListLY,, PredFlagLY[ x ][ y ], MvLY[ x ][ y ], and RefIdxLY[ x ][ y ], respectively.
  + When predFlagLYIvRef[ xIvRefPb ][ yIvRefPb ] is equal to 1, the following applies for each i from 0 to num\_ref\_idx\_lX\_active\_minus1, inclusive:
    - * When PicOrderCnt( refPicListLYIvRef[ refIdxLYIvRef[ xIvRefPb ][ yIvRefPb ] ]) is equal to PicOrderCnt( RefPicListLX[ i ] ) and availableFlagLXInterView is equal to 0, the following applies.

spMvLXInterView[blkX][blkY] [ 0 ] = mvLYIvRef[ xIvRefPb ][ yIvRefPb ] [ 0 ] (‑126)  
spMvLXInterView[blkX][blkY] [ 1 ] = mvLYIvRef[ xIvRefPb ][ yIvRefPb ] [ 1 ] (‑127)

spRefIdxLX[blkX][blkY] = i (‑128)

availableFlagLXInterView = 1

* If (spRefIdxL0[blkX][blkY]!=-1 || spRefIdxL1[blkX][blkY]!=-1)
  + if (availableFlagL0InterView == -1 && availableFlagL1InterView == -1)
    - * For each block location (i = 0..blkX-1, j=0..blkY-1), the following applies
        + spMvLXInterView[i][j][0] = spMvLXInterView[blkX][blkY][0]
        + spMvLXInterView[i][j][1] = spMvLXInterView[blkX][blkY][1]
        + spRefIdxLX[i][j] = spRefIdxLX[blkX][blkY]
        + spPredFlagLX[i][j] = spPredFlagLX[blkX][blkY]
      * mvLXInterView[0] = spMvLXInterView[blkX][blkY][0]
      * mvLXInterView[1] = spMvLXInterView[blkX][blkY][1]
      * refIdxLX = spRefIdxLX[blkX][blkY]
  + tempMvLX[0] = spMvLXInterView[blkX][blkY][0]
  + tempMvLX[1] = spMvLXInterView[blkX][blkY][1]
  + tempRefIdxLX = spRefIdxLX[blkX][blkY]
  + tempPredFlagLX = spPredFlagLX[blkX][blkY]
* Else if (tempRefIdxL0!=-1 || tempRefIdxL1!=-1)
  + spMvLXInterView[blkX][blkY][0] = tempMvLX[0]
  + spMvLXInterView[blkX][blkY][1] = tempMvLX[1]
  + spRefIdxLX[blkX][blkY] = tempRefIdxLX
  + spPredFlagLX[blkX][blkY] = tempPredFlagLX

The flag availableFlagLXInterView is set to refIdxLX!=-1