# Draft Text Specification

The proposed text changes are based on the document JCTVC-P1005-v1.doc for the Intra BC prediction in JCTVC-Q0114. The changes are marked in yellow.

**7.3.8.5 Coding unit syntax**

|  |  |
| --- | --- |
| coding\_unit( x0, y0, log2CbSize ) { | Descriptor |
| …. |  |
| } else if( intra\_bc\_flag[ x0 ][ y0 ] ) { |  |
| **intra\_bc\_bvp\_flag**[ x0 ][ y0 ] | ae(v) |
| mvd\_coding( x0, y0, 2) |  |
| if( PartMode = = PART\_2NxN ) |  |
| **intra\_bc\_bvp\_flag**[ x0 ][ y0+ ( nCbS / 2) ] | ae(v) |
| mvd\_coding( x0, y0 + ( nCbS / 2 ), 2) |  |
| else if( PartMode = = PART\_Nx2N ) |  |
| **intra\_bc\_bvp\_flag**[ x0 + ( nCbS / 2 ) ][ y0 ] | ae(v) |
| mvd\_coding( x0 + ( nCbS / 2 ), y0, 2) |  |
| else if( PartMode = = PART\_NxN ) { |  |
| **intra\_bc\_bvp\_flag**[ x0 + ( nCbS / 2 ) ][ y0 ] | ae(v) |
| mvd\_coding( x0 + ( nCbS / 2 ), y0, 2) |  |
| **intra\_bc\_bvp\_flag**[ x0 ][ y0+ ( nCbS / 2) ] | ae(v) |
| mvd\_coding( x0, y0 + ( nCbS / 2 ), 2) |  |
| **intra\_bc\_bvp\_flag**[ x0 + ( nCbS / 2 ) ][ y0+ ( nCbS / 2) ] | ae(v) |
| mvd\_coding( x0 + ( nCbS / 2 ), y0 + ( nCbS / 2 ), 2) |  |
| } |  |
| … |  |
| } |  |

**intra\_bc\_bvp\_flag**[ x0 ][ y0 ] equal to 1 specifies that above block vector is used as the block vector predictor. intra\_bc\_bvp\_flag[ x0 ][ y0 ] equal to 0 specifies that left block vector is used as the block vector predictor. When not present, the value of intra\_bc\_bvp\_flag[ x0 ][ y0 ] is inferred to be equal to 0. The array indices x0, y0 specify the location ( x0, y0 ) of the top-left luma sample of the considered coding block relative to the top-left luma sample of the picture.

**8.4.4 Derivation process for block vector components in intra block copying prediction mode**

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 variable log2CbSize specifying the size of the current luma coding block.

Output of this process is the (nCbS)x(nCbX) array of block vectors bvIntra.

The variables nCbS, nCbSw, nCbSh are derived as follows:

nCbS = 1  <<  log2CbSize (8‑25)

nPbSw = nCbS / ( PartMode = = PART\_2Nx2N | | PartMode = = PART\_2NxN ? 1 : 2 ) (8‑25)

nPbSh = nCbS / ( PartMode = = PART\_2Nx2N | | PartMode = = PART\_Nx2N ? 1 : 2 ) (8‑25)

The variable BvpIntra[ compIdx ] specifies a block vector predictor. The horizontal block vector component is assigned compIdx = 0 and the vertical block vector component is assigned compIdx = 1.

Depending upon PartMode, the variable numPartitions is derived as follows:

– If PartMode is equal to PART\_2Nx2N, numPartitions is set equal to 1.

– Otherwise, if PartMode is equal to either PART\_2NxN or PART\_Nx2N, numPartitions is set equal to 2.

– Otherwise (PartMode is equal to PART\_NxN), numPartitions is set equal to 4.

The array of block vectors bvIntra is derived by the following ordered steps, for the variable blkIdx proceeding over the values 0..( numPartitions − 1 ):

1. The variable blkInc is set equal to ( PartMode = = PART\_2NxN ? 2 : 1 ).
2. The variable xPb is set equal to xCb + nPbSw \* ( blkIdx \* blkInc % 2 ).
3. The variable yPb is set equal to yCb + nPbSh \* ( blkIdx / 2 )
4. The following ordered steps apply, for the variable compIdx proceeding over the values 0..1:
5. Depending upon the number of times this process has been invoked for the current coding tree unit, the following applies:

* ~~If this process is invoked for the first time for the current coding tree unit, bvIntra[ xPb ][ yPb ][ compIdx ] is derived as follows:~~

~~bvIntra[ xPb ][ yPb ][ 0 ] = BvdIntra[ xPb ][ yPb ][ 0 ] − nCbS (8‑25)~~

~~bvIntra[ xPb ][ yPb ][ 1 ] = BvdIntra[ xPb ][ yPb ][ 1 ] (8‑25)~~

* ~~Otherwise, bvIntra[ xPb ][ yPb ][ compIdx ] is derived as follows:~~

~~bvIntra[ xPb ][ yPb ][ 0 ] = BvdIntra[ xPb ][ yPb ][ 0 ] + BvpIntra[ 0 ] (8‑25)~~

~~bvIntra[ xPb ][ yPb ][ 1 ] = BvdIntra[ xPb ][ yPb ][ 1 ] + BvpIntra[ 1 ] (8‑25)~~

* The variables BvpIntraL[ compIdx ] specifies the left neighboring block vector predictor. The horizontal block vector component is assigned compIdx = 0 and the vertical block vector component is assigned compIdx = 1. The variable BvpIntraLFlag specifies the availability flags of the left neighboring prediction units. BvpIntraL[ compIdx ] and BvpIntraLFlag are derived as follows:
  + BvpIntraL[ compIdx ] and BvpIntraLFlag are set equal to 0;
  + If the derivation process for z-scan order block availability as specified in subclause 6.4.1 is invoked with ( xCurr, yCurr ) set equal to ( xCb, yCb ) and the neighbouring luma location ( xNbY, yNbY ) set equal to ( xPb ‑ 1 , yPb ) as inputs, the output is equal to TRUE, and intra\_bc\_flag[ ( xPb ‑ 1  ][ yPb ] is equal to 1, then BvpIntraLFlag is set to 1, and

BvpIntraL[compIdx] = bvIntra[xPb ‑ 1][yPb]

* + Otherwise, If the derivation process for z-scan order block availability as specified in subclause 6.4.1 is invoked with ( xCurr, yCurr ) set equal to ( xCb, yCb ) and the neighbouring luma location ( xNbY, yNbY ) set equal to ( xPb ‑ 1 , yPb + nPbSh ) as inputs, the output is equal to TRUE, and intra\_bc\_flag[( xPb ‑ 1 ][  yPb + nPbSh ] is equal to 1, then BvpIntraLFlag is set to 1, and

BvpIntraL[compIdx] = bvIntra[xPb ‑ 1][yPb + nPbSh ]

* The variables BvpIntraA[ compIdx ] specifies the above neighboring block vector predictor. The horizontal block vector component is assigned compIdx = 0 and the vertical block vector component is assigned compIdx = 1. The variable BvpIntraAFlag specifies the availability flags of the above neighboring prediction units. BvpIntraA[ compIdx ] and BvpIntraAFlag are derived as follows:
  + BvpIntraA[ compIdx ] and BvpIntraAFlag are set equal to 0;
  + If the derivation process for z-scan order block availability as specified in subclause 6.4.1 is invoked with ( xCurr, yCurr ) set equal to ( xCb, yCb ) and the neighbouring luma location ( xNbY, yNbY ) set equal to ( xPb  , yPb - 1 ) as inputs, the output is equal to TRUE, and intra\_bc\_flag[ xPb  ][ yPb - 1 ] is equal to 1, then BvpIntraLFlag is set to 1, and

BvpIntraA[compIdx] = bvIntra[ xPb ][yPb - 1 ]

* + Otherwise, If the derivation process for z-scan order block availability as specified in subclause 6.4.1 is invoked with ( xCurr, yCurr ) set equal to ( xCb, yCb ) and the neighbouring luma location ( xNbY, yNbY ) set equal to ( xPb + nPbSw , yPb - 1 ) as inputs, the output is equal to TRUE, and intra\_bc\_flag[ xPb + nPbSw ][ yPb - 1 ] is equal to 1, then BvpIntraLFlag is set to 1, and

BvpIntraA[compIdx] = bvIntra[ xPb + nPbSw ][ yPb - 1 ]

* BvpIntra[ compIdx ] is derived as follows:
  + If BvpIntraLFlag is equal to 1, and BvpIntraAFlag is equal to 1, then

If intra\_bc\_bvp\_flag[ xPb ][ yPb ] is equal to 0, BvpIntra [compIdx] = BvpIntraL[compIdx];

Otherwise, BvpIntra [compIdx] = BvpIntraA[compIdx];

* + Otherwise, if BvpIntraLFlag is equal to 1, and BvpIntraAFlag is equal to 0, then

If intra\_bc\_bvp\_flag[ xPb ][ yPb ] is equal to 0, BvpIntra [compIdx] = BvpIntraL[compIdx];

Otherwise, BvpIntra [0] = - 2 \* nPbSw, BvpIntra [1] = 0;

* + Otherwise, if BvpIntraLFlag is equal to 0, and BvpIntraAFlag is equal to 1, then

If intra\_bc\_bvp\_flag[ xPb ][ yPb ] is equal to 0, BvpIntra [compIdx] = BvpIntraA[compIdx];

Otherwise, BvpIntra [0] = - 2 \* nPbSw, BvpIntra [1] = 0;

* + Otherwise, if BvpIntraLFlag is equal to 0, and BvpIntraAFlag is equal to 0, then

If intra\_bc\_bvp\_flag[ xPb ][ yPb ] is equal to 0, BvpIntra [0] = - 2 \* nPbSw, BvpIntra [1] = 0;

Otherwise, BvpIntra [0] = 2 \* nPbSw, BvpIntra [1] = 0;

* The bvIntra[ xPb ][ yPb ][ compIdx ] is derived as follows:

bvIntra[ xPb ][ yPb ][ 0 ] = BvdIntra[ xPb ][ yPb ][ 0 ] + BvpIntra [ 0 ] (8‑25)  
bvIntra[ xPb ][ yPb ][ 1 ] = BvdIntra[ xPb ][ yPb ][ 1 ] + BvpIntra [ 1 ] (8‑25)

1. ~~The value of BvpIntra[ compIdx ] is updated to be equal to bvIntra[ xPb ][ yPb ][ compIdx ].~~