# Proposed text

**7.3.8.5 Coding unit syntax**

|  |  |
| --- | --- |
| **…** |  |
| } else if( intra\_bc\_flag[ x0 ][ y0 ] ) { |  |
| **part\_mode** | ae(v) |
| mvd\_coding( x0, y0, 2) |  |
| if( PartMode == Part\_2NxN ) |  |
| mvd\_coding( x0, y0 + ( nCbS/2 ), 2 ) |  |
| else if( PartMode == Part\_Nx2N ) |  |
| mvd\_coding( x0 + ( nCbS/2 ), y0, 2 ) |  |
| } else if( PartMode = = PART\_2NxnU ) { |  |
| mvd\_coding( x0 + ( nCbS/2 ), y0, 2 ) |  |
| } else if( PartMode = = PART\_2NxnD ) { |  |
| mvd\_coding( x0 + ( nCbS/2 ), y0, 2 ) |  |
| } else if( PartMode = = PART\_nLx2N) { |  |
| mvd\_coding( x0 + ( nCbS/2 ), y0, 2 ) |  |
| } else if( PartMode = = PART\_nRx2N) { |  |
| mvd\_coding( x0 + ( nCbS/2 ), y0, 2 ) |  |
| else if( PartMode == Part\_NxN ) { |  |
| mvd\_coding( x0 + ( nCbS/2 ), y0, 2 ) |  |
| mvd\_coding( x0, y0 + ( nCbS/2 ), 2 ) |  |
| mvd\_coding( x0 + ( nCbs/2 ), y0 + ( nCbS/2 ), 2 ) |  |
| } |  |
| } else { |  |
| **…** |  |

**part\_mode** specifies partitioning mode of the current coding unit. The semantics of part\_mode depend on CuPredMode[ x0 ][ y0 ]. The variables PartMode and IntraSplitFlag are derived from the value of part\_mode as defined in Table 7‑10.

The value of part\_mode is restricted as follows:

* If CuPredMode[ x0 ][ y0 ] is equal to MODE\_INTRA, the following applies
* If intra\_bc\_flag[x0][y0] is equal to 1, part\_mode shall be in the range of 0 to 7, inclusive. The mode is set same as MODE\_INTER
* Otherwise (if intra\_bc\_flag[x0][y0] is equal to 0), part\_mode shall be equal to 0 or 1.
* Otherwise (CuPredMode[ x0 ][ y0 ] is equal to MODE\_INTER), the following applies:

…

* Table 7‑10 – Name association to prediction mode and partitioning type

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CuPredMode [ x0 ][ y0 ]** | **intra\_bc\_flag [ x0 ][ y0 ]** | **part\_mode** | **IntraSplitFlag** | **PartMode** |
| MODE\_INTRA | 0 | 0 | 0 | PART\_2Nx2N |
| 1 | 1 | PART\_NxN |
| 1 | SAME as MODE\_INTER | | |
| MODE\_INTER | - | 0 | 0 | PART\_2Nx2N |
| 1 | 0 | PART\_2NxN |
| 2 | 0 | PART\_Nx2N |
| 3 | 0 | PART\_NxN |
| 4 | 0 | PART\_2NxnU |
| 5 | 0 | PART\_2NxnD |
| 6 | 0 | PART\_nLx2N |
| 7 | 0 | PART\_nRx2N |

#### 9.3.3.5 Binarization process for part\_mode

Inputs to this process are a request for a binarization for the syntax element part\_mode a luma location ( xCb, yCb ), specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture, and a variable log2CbSize specifying the current luma coding block size.

Output of this process is the binarization of the syntax element.

The binarization for the syntax element part\_mode is specified in Table 9‑36 depending on the values of CuPredMode[ xCb ][ yCb ] and log2CbSize.

**Table 9‑36 – Binarization for part\_mode**

Table 9‑36 – Binarization for part\_mode

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **CuPredMode** [ xCb ][ yCb ] | **intra\_bc\_flag** [ xCb ][ yCb ] | **part\_mode** | **PartMode** | **Bin string** | | | |
| log2CbSize >  MinCbLog2SizeY | | log2CbSize  = =  MinCbLog2SizeY | |
| !amp\_enabled\_flag | amp\_enabled\_flag | log2CbSize  = =  3 | log2CbSize > 3 |
| MODE\_INTRA | 0 | 0 | PART\_2Nx2N | - | - | 1 | 1 |
| 1 | PART\_NxN | - | - | 0 | 0 |
| 1 | SAME as MODE\_INTER | | | | | |
| MODE\_INTER |  | 0 | PART\_2Nx2N | 1 | 1 | 1 | 1 |
|  | 1 | PART\_2NxN | 01 | 011 | 01 | 01 |
|  | 2 | PART\_Nx2N | 00 | 001 | 00 | 001 |
|  | 3 | PART\_NxN | - | - | - | 000 |
|  | 4 | PART\_2NxnU | - | 0100 | - | - |
|  | 5 | PART\_2NxnD | - | 0101 | - | - |
|  | 6 | PART\_nLx2N | - | 0000 | - | - |
|  | 7 | PART\_nRx2N | - | 0001 | - | - |

**8.4.1** **General decoding process for coding units coded in intra prediction mode**

…

Otherwise (pcm\_flag[ xCb ][ yCb ] is equal to 0 and IntraSplitFlag is equal to 1), for the variable blkIdx proceeding over the values 0..3, the following ordered steps apply:

1. The variable xPb is set equal to xCb + ( nCbS  >>  1 ) \* ( blkIdx % 2 ).
2. The variable yPb is set equal to yCb + ( nCbS  >>  1 ) \* ( blkIdx / 2 ).
3. The derivation process for the intra prediction mode as specified in subclause 8.4.2 is invoked with the luma location ( xPb, yPb ) as input.
4. The general decoding process for intra blocks as specified in subclause 8.4.4.1 is invoked with the luma location ( xPb, yPb ), the variable log2TrafoSize set equal to log2CbSize − 1, the variable trafoDepth set equal to 1, the variable predModeIntra set equal to IntraPredModeY[ xPb ][ yPb ], the variable predModeIntraBc set equal to intra\_bc\_flag[xCb][yCb], the variable bvIntra set equal to bvIntra[xPb][yPb], and the variable cIdx set equal to 0 as inputs, and the output is a modified reconstructed picture before deblocking filtering.

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

Inputs to this process are:

* 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 log2TrafoSize specifying the size of the current luma coding block.

Output of this process is the block vector bvIntra.

The variable BvpIntra[ xPb ][ yPb ] [ 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.

The block vector bvIntra is derived by the following ordered steps, for the variable compIdx proceeding over the values 0..1:

1. 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 ][ compIdx ] = BvdIntra[ xPb ][ yPb ][ compIdx ] (8‑25)

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

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

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

It is a requirement of bitstream conformance that all of the following conditions are true:

– The value of bvIntra[ xPb ][ yPb ][ 0 ] shall be greater than or equal to – ( xPb % CtbSizeY + CtbSizeY ).

– The value of bvIntra[ xPb ][ yPb ][ 1 ] shall be greater than or equal to – ( yPb % CtbSizeY ).

– When the derivation process for z-scan order block availability as specified in subclause 6.4.1 is invoked with ( xCurr, yCurr ) set equal to ( xPb, yPb ) and the neighbouring luma location ( xNbY, yNbY ) set equal to ( xPb + bvIntra[ xPb ][ yPb ][ 0 ], yPb + bvIntra[ xPb ][ yPb ][ 1 ] ) as inputs, the output shall be equal to TRUE.

– When the derivation process for z-scan order block availability as specified in subclause 6.4.1 is invoked with ( xCurr, yCurr ) set equal to ( xPb, yPb ) and the neighbouring luma location ( xNbY, yNbY ) set equal to ( xPb + bvIntra[ xPb ][ yPb ][ 0 ] +  ( nCbS >> ( PartMode ! = PART\_NxN ? 0 : 1 ) ) − 1, yPb + bvIntra[ xPb ][ yPb ][ 1 ] + ( nCbS >> ( PartMode ! = PART\_NxN ? 0 : 1 ) ) – 1 ) as inputs, the output shall be equal to TRUE.

– One or both of the following conditions shall be true:

– bvIntra[ xPb ][ yPb ][ 0 ] + ( nCbS >> ( PartMode ! = PART\_NxN? 0 : 1 ) ) <= 0

– bvIntra[ xPb ][ yPb ][ 1 ] + ( nCbS >> ( PartMode ! = PART\_NxN? 0 : 1 ) ) <= 0

**8.4.5.2.7 Specification of intra block copying prediction mode**

Inputs to this process are:

– a sample location ( xTbCmp, yTbCmp ) specifying the top-left sample of the current transform block relative to the top‑left sample of the current picture,

– a variable nCbS specifying the coding block size,

– a variable nTbS specifying the transform block size,

– a variable bvIntra specifying the block copying vector,

– a variable cIdx specifying the colour component of the current block.

Output of this process is the predicted samples predSamples[ x ][ y ], with x, y = 0..nTbS − 1.

* The variable bv representing the block vector for prediction in full-sample units is derived as follows:

– If cIdx is larger than 0 and nTbs is equal to 4 and chroma\_format\_idc is equal to 1,

bv[ 0 ] = max( bvIntra[ xCb +  ( nCbs >> 1 ) ][ yCb +  ( nCbs >> 1 )][ 0 ],   
 – ( xCb % CtbSizeY + CtbSizeY )) >> ( SubWidthC – 1 ) )

bv[ 1 ] = max( bvIntra[ xCb +  ( nCbs >> 1 ) ][ yCb +  ( nCbs >> 1 )][ 1 ],   
 – ( yCb % CtbSizeY)) >> ( SubWidthC – 1 ) )

– Otherwise, if cIdx is larger than 0 and nTbs is equal to 4 and chroma\_format\_idc is equal to 2 and yTbCmp is smaller than yCb + ( nCbs >> 1 ),

bv[ 0 ] = max( bvIntra[ xCb +  ( nCbs >> 1 ) ][ yCb )][ 0 ], – ( xCb % CtbSizeY + CtbSizeY )) >>  
 ( SubWidthC – 1 ) )

bv[ 1 ] = bvIntra[ xCb +  ( nCbs >> 1 ) ][ yCb ][ 1 ] >> ( SubWidthC – 1 ) )

– Otherwise, if cIdx is larger than 0 and nTbs is equal to 4 and chroma\_format\_idc is equal to 2 and yTbCmp is not smaller than yCb + (nCbs>>1)

bv[ 0 ] = max( bvIntra[ xCb +  ( nCbs >> 1 ) ][ yCb +  ( nCbs >> 1 )][ 0 ],   
 – ( xCb % CtbSizeY + CtbSizeY )) >> ( SubWidthC – 1 ) )

bv[ 1 ] = bvIntra[ xCb +  ( nCbs >> 1 ) ][ yCb +  ( nCbs >> 1 )][ 1 ] >> ( SubWidthC – 1 ) )

– Otherwise,

bv[ 0 ] = bvIntra[ xTbCmp ][ yTbCmp ][ 0 ] >> ( ( ( cIdx = = 0 ) ? 1 : SubWidthC ) − 1 ) (8‑63)

bv[ 1 ] = bvIntra[ xTbCmp ][ yTbCmp ][ 1 ] >> ( ( ( cIdx = = 0 ) ? 1 : SubHeightC ) − 1 ) (8‑64)

* The (nTbS)x(nTbS) array of predicted samples samples, with x, y = 0..nTbS − 1, is derived as follows:

– The reference sample location (xRefCmp, yRefCmp ) is specified by:

( xRefCmp, yRefCmp ) = ( xTbCmp + x + bv[ 0 ], yTbCmp + y + bv[ 1 ] ) (8‑65)

* Each sample at the location ( xRefCmp, yRefCmp ) is assigned to predSamples[ x ][ y ].