# Draft Text Specification

The proposed text changes are based on the document JCTVC-M1005-v2.doc for the MV coding method 2 in JCTVC-N0256. The changes are marked in yellow.

#### 7.3.8.5 Coding unit syntax

|  |  |
| --- | --- |
| coding\_unit( x0, y0, log2CbSize ) { | Descriptor |
| if( transquant\_bypass\_enabled\_flag ) |  |
| **cu\_transquant\_bypass\_flag** | ae(v) |
| if( slice\_type != I ) |  |
| **cu\_skip\_flag**[ x0 ][ y0 ] | ae(v) |
| nCbS = ( 1  <<  log2CbSize ) |  |
| if( cu\_skip\_flag[ x0 ][ y0 ] ) |  |
| prediction\_unit( x0, y0, nCbS, nCbS ) |  |
| else { |  |
| **intra\_mc\_flag**[ x0 ][ y0 ] | ae(v) |
| if( intra\_mc\_flag[ x0 ][ y0 ] ) { |  |
| mvd\_coding( x0, y0, 2 ) |  |
| } else { |  |
| if( slice\_type != I ) |  |
| **pred\_mode\_flag** | ae(v) |
| if( CuPredMode[ x0 ][ y0 ] != MODE\_INTRA | | log2CbSize = = MinCbLog2SizeY ) |  |
| **part\_mode** | ae(v) |
| if( CuPredMode[ x0 ][ y0 ] = = MODE\_INTRA ) { |  |
| … |  |
| } |  |
| } |  |
| if( !pcm\_flag[ x0 ][ y0 ] ) { |  |
| if( ( CuPredMode[ x0 ][ y0 ] != MODE\_INTRA &&   !( PartMode = = PART\_2Nx2N && merge\_flag[ x0 ][ y0 ] ) ) || CuPredMode[ x0 ][ y0 ] == MODE\_INTRA\_MC ) |  |
| **rqt\_root\_cbf** | ae(v) |
| … |  |
| } |  |
| } |  |
| } |  |

**7.4.9.5 Coding unit semantics**

**…**

**intra\_mc\_flag[ x0 ][ y0 ]** equal to 1 specifies that the current coding unit is coded in intra motion compensation mode. intra\_mc\_flag[ x0 ][ y0 ]equal to 0 specifies that the current coding unit is coded using MODE\_INTRA, MODE\_INTER or MODE\_SKIP.

When intra\_mc\_flag[ x0 ][ y0 ] is not present, it is inferred to be equal to 0.

**pred\_mode\_flag** equal to 0 specifies that the current coding unit is coded in inter prediction mode. pred\_mode\_flag equal to 1 specifies that the current coding unit is coded in intra prediction mode. The variable CuPredMode[ x ][ y ] is derived as follows for x = x0..x0 + nCbS − 1 and y = y0..y0 + nCbS − 1:

* If pred\_mode\_flag is equal to 0, CuPredMode[ x ][ y ] is set equal to MODE\_INTER.
* Otherwise (pred\_mode\_flag is equal to 1), CuPredMode[ x ][ y ] is set equal to MODE\_INTRA.

When pred\_mode\_flag is not present, the variable CuPredMode[ x ][ y ] is derived as follows for x = x0..x0 + nCbS − 1 and y = y0..y0 + nCbS − 1:

* If intra\_mc\_flag[ x0 ][ y0 ] is equal to 1, CuPredMode[ x ][ y ] is inferred to be equal to MODE\_INTRA\_MC.
* Otherwise, if slice\_type is equal to I, CuPredMode[ x ][ y ] is inferred to be equal to MODE\_INTRA.
* Otherwise (slice\_type is equal to P or B), when cu\_skip\_flag[ x0 ][ y0 ] is equal to 1, CuPredMode[ x ][ y ] is inferred to be equal to MODE\_SKIP.

**7.4.9.5 Coding unit semantics**

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

|  |  |  |  |
| --- | --- | --- | --- |
| **CuPredMode[ x0 ][ y0 ]** | **part\_mode** | **IntraSplitFlag** | **PartMode** |
| MODE\_INTRA | 0 | 0 | PART\_2Nx2N |
| 1 | 1 | PART\_NxN |
| 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 |
| MODE\_INTRA\_MC | 0 | 0 | PART\_2Nx2N |

**7.4.9.9 Motion vector difference semantics**

…

* If refList is equal to 0, MvdL0[ x0 ][ y0 ][ compIdx ] is set equal to lMvd[ compIdx ] for compIdx = 0..1.
* Otherwise if refList is equal to 1, MvdL1[ x0 ][ y0 ][ compIdx ] is set equal to lMvd[ compIdx ] for compIdx = 0..1.

**8.x Decoding process for coding units coded in intra motion compensation mode**

**8.x.1 General decoding process for coding units coded in intra motion compensation mode**

Inputs to this process are:

– 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,

– a variable log2CbSize specifying the size of the current coding block.

Output of this process is a modified reconstructed picture before deblocking filtering.

The derivation process for quantization parameters as specified in subclause 8.6.1 is invoked with the luma location ( xCb, yCb ) as input.

The variable nCbSL is set equal to 1  <<  log2CbSize. When ChromaArrayType is not equal to 0, the variable nCbSwC is set equal to 1  <<  ( log2CbSize ) / SubWidthC and the variable nCbShC is set equal to ( 1  <<  log2CbSize ) / SubHeightC.

The decoding process for coding units coded in intra motion compensation mode consists of following ordered steps:

1. The intra motion compensation prediction process as specified in subclause 8.x.2 is invoked with the luma location ( xCb, yCb ) and the luma coding block size log2CbSize as inputs, and the output is the array predSamplesL and when ChromaArrayType is not equal to 0, the arrays predSamplesCb, and predSamplesCr.
2. The decoding process for the residual signal of coding units coded in inter prediction mode specified in subclause 8.5.4 is invoked with the luma location ( xCb, yCb ) and the luma coding block size log2CbSize as inputs, and the output is the array resSamplesL and when ChromaArrayType is not equal to 0, the arrays resSamplesCb, and resSamplesCr.
3. The reconstructed samples of the current coding unit are derived as follows:

– The picture reconstruction process prior to in-loop filtering for a colour component as specified in subclause 8.6.5 is invoked with the luma coding block location ( xCb, yCb ), the variable nCurrSw set equal to nCbSwL, the variable nCurrSh set equal to nCbShL, the variable cIdx set equal to 0, the (nCbSL)x(nCbSL) array predSamples set equal to predSamplesL, and the (nCbSL)x(nCbSL) array resSamples set equal to resSamplesL as inputs.

– When ChromaArrayType is not equal to 0, the picture reconstruction process prior to in-loop filtering for a colour component as specified in subclause 8.6.5 is invoked with the chroma coding block location ( xCb / SubWidthC, yCb / SubHeightC ), the variable nCurrSw set equal to nCbSwC, the variable nCurrSh set equal to nCbShC, the variable cIdx set equal to 1, the (nCbSwC)x(nCbShC) array predSamples set equal to predSamplesCb, and the (nCbSwC)x(nCbShC) array resSamples set equal to resSamplesCb as inputs.

– When ChromaArrayType is not equal to 0, the picture reconstruction process prior to in-loop filtering for a colour component as specified in subclause 8.6.5 is invoked with the chroma coding block location ( xCb / SubWidthC, yCb / SubHeightC ), the variable nCurrSw set equal to nCbSwC, the variable nCurrSh set equal to nCbShC, the variable cIdx set equal to 2, the (nCbSwC)x(nCbShC) array predSamples set equal to predSamplesCr, and the (nCbSwC)x(nCbShC) array resSamples set equal to resSamplesCr as inputs.

**8.x.2** **Intra motion compensation prediction process**

This process is invoked when decoding coding unit whose CuPredMode[ xCb ][ yCb ] is equal to MODE\_INTRA\_MC.

Inputs to this process are:

– 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,

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

Outputs of this process are:

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

– when ChromaArrayType is not equal to 0, an (nCbSwC)x(nCbShC) array predSamplesCb of chroma prediction samples for the component Cb, where nCbSwC and nCbShC are derived as specified below,

– when ChromaArrayType is not equal to 0, an (nCbSwC)x(nCbShC) array predSamplesCr of chroma prediction samples for the component Cr, where nCbSwC and nCbShC are derived as specified below.

The variable nCbSL is set equal to 1  <<  log2CbSize. When ChromaArrayType is not equal to 0, the variable nCbSwC is set equal to nCbSL / SubWidthC and the variable nCbShC is set equal to nCbSL / SubHeightC.

The variable nCbS1L is set equal to nCbSL  >>  1.

The decoding process for prediction units in intra motion compensation mode as specified in subclause 8.x.3 is invoked with the luma location ( xCb, yCb ), the luma location ( xBl, yBl ) set equal to ( 0, 0 ), the size of the luma coding block nCbSL, the width of the luma prediction block nPbW set equal to nCbSL, the height of the luma prediction block nPbH set equal to nCbSL, and a partition index partIdx set equal to 0 as inputs, and the outputs are an (nCbSL)x(nCbSL) array predSamplesL and when ChromaArrayType is not equal to 0, two (nCbSwC)x(nCbShC) arrays predSamplesCb and predSamplesCr.

**8.x.3 Decoding process for prediction units in intra motion compensation prediction mode**

Inputs to this process are:

– 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,

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

Outputs of this process are:

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

– when ChromaArrayType is not equal to 0, an (nCbSwC)x(nCbShC) array predSamplesCb of chroma prediction samples for the component Cb, where nCbSwC and nCbShC are derived as specified below,

– when ChromaArrayType is not equal to 0, an (nCbSwC)x(nCbShC) array predSamplesCr of chroma prediction samples for the component Cr, where nCbSwC and nCbShC are derived as specified below.

The variable nCbSL is set equal to nCbS. When ChromaArrayType is not equal to 0, the variable nCbSwC is set equal to nCbS / SubWidthC and the variable nCbShC is set equal to nCbS / SubHeightC.

The decoding process for prediction units in intra motion compensation prediction mode consists of the following ordered steps:

1. The derivation process for intra motion vector components in subclause 8.5.x.2 is invoked with the luma coding block location ( xCb, yCb ), the luma coding block size block nCbS, mvLI, when ChromaArrayType is not equal to 0, the chroma motion vectors mvCLI.
2. The decoding process for intra motion compensation sample prediction as specified in subclause 8.5.x.3 is invoked with the luma coding block location ( xCb, yCb ), the luma coding block size block nCbS, the luma prediction block width nPbW, the luma prediction block height nPbH, mvLI, when ChromaArrayType is not equal to 0, the chroma motion vectors mvCLI, and the intra motion compensation prediction samples (predSamples) that are an (nCbSL)x(nCbSL) array predSamplesL of prediction luma samples and when ChromaArrayType is not equal to 0, two (nCbSwC)x(nCbShC) arrays predSamplesCr and predSamplesCr of prediction chroma samples, one for each of the chroma components Cb and Cr, as outputs.

**8.5.x.2** **Derivation process for intra motion vector components**

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 nCbS specifying the size of the current luma coding block,

Outputs of this process are:

* the luma motion vectors mvLI, which is in integer-luma-sample units,
* when ChromaArrayType is not equal to 0, the chroma motion vectors mvCLI.

For the derivation of the variable mvLI, the following applies:

* mvLI[0] = lMvd[0], mvLI[1] = lMvd[1].

When ChromaArrayType is not equal to 0, for the derivation of variable mvCLI, the following applies:

* mvCLI [0] = mvLI[0] / subWidthC; mvCLI[1] = mvLI[1] / subHeightC;

**8.5.x.3** **Derivation process for intra motion compensation prediction samples**

Inputs to this process are:

– 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,

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

– the luma motion vectors mvLI,

– when ChromaArrayType is not equal to 0, the chroma motion vectors mvCLI,

* a bit depth of samples, bitDepth.

Outputs of this process are:

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

– when ChromaArrayType is not equal to 0, an (nCbSwC)x(nCbShC) array preSamplesCb of chroma prediction samples for the component Cb, where nCbSwC and nCbShC are derived as specified below,

– when ChromaArrayType is not equal to 0, an (nCbSwC)x(nCbShC) array predSamplesCr of chroma residual samples for the component Cr, where nCbSwC and nCbShC are derived as specified below.

The variable nCbSL is set equal to nCbS. When ChromaArrayType is not equal to 0, the variable nCbSwC is set equal to nCbS / SubWidthC and the variable nCbShC is set equal to nCbS / SubHeightC. The reference picture consisting of current reconstructed picture.

Variables shift1, shift2, offset1, and offset2 are derived as follows:

– The variable shift1 is set equal to 14 − bitDepth and the variable shift2 is set equal to 15 − bitDepth.

– The variable offset1 is derived as follows:

* If shift1 is greater than 0, offset1 is set equal to 1  <<  ( shift1 − 1 ).
* Otherwise (shift1 is equal to 0), offset1 is set equal to 0.

– The variable offset2 is set equal to 1  <<  ( shift2 − 1 ).

The prediction samples predSamples[ x ][ y ] are derived as follows:

predSamples[ x ][ y ] = Clip3( 0, ( 1  <<  bitDepth ) − 1, ( predSamplesLI[ x ][ y ] + offset1 )  >>  shift1 )

**9.3.2.2 Initialization process for context variables**

**Table 4‑1 – Association of ctxIdx and syntax elements for each initializationType in the initialization process**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Syntax element | ctxIdxTable | initType | | |
| 0 | 1 | 2 |
| coding\_unit() | intra\_mc\_flag | Table 9-xx | 0..2 | 3..5 | 6..8 |

**Table 9-xx. Values of variable initValue for intra\_mc\_flag ctxIdx**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Initialization variable | intra\_mc\_flag ctxIdx | | | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| initValue | 185 | 185 | 201 | 197 | 197 | 185 | 197 | 197 | 185 |

**9.3.3 Binarization process**

**9.3.3.1 General**

**Table 4‑2 – Syntax elements and associated binarizations**

| Syntax structure | Syntax element | Binarization | |
| --- | --- | --- | --- |
| Process | Input parameters |
| coding\_uint( ) | intra\_mc\_flag | FL | cMax = 1 |

**9.3.4.2 Derivation process for ctxTable, ctxIdx and bypassFlag**

**9.3.4.2.1 General**

**Table 4‑3 – Assignment of ctxInc to syntax elements with context coded bins**

| Syntax element | binIdx | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | >= 5 |
| intra\_mc\_flag | 0,1,2 | na | na | na | na | na |

**9.3.4.2.2 Derivation process of ctxInc using left and above syntax elements**

**Table 4‑4 – Specification of ctxInc using left and above syntax elements**

|  |  |  |  |
| --- | --- | --- | --- |
| Syntax element | condL | condA | ctxInc |
| intra\_mc\_flag[ x0 ][ y0 ] | intra\_mc\_flag [ xNbL ][ yNbL ] | intra\_mc\_flag [ xNbA ][ yNbA ] | ( condL  &&  availableL ) + ( condA  &&  availableA ) |