# Draft Text Specification

The proposed text changes are based on the document JCTVC-M1005-v2.doc. The changes are marked in yellow. It is suggested to replace subsection 8.4.4.2.1 and additionally add section 8.4.4.2.7.

**7.3.8.11 Residual coding syntax**

|  |  |
| --- | --- |
| residual\_coding( x0, y0, log2TrafoSize, cIdx ) { | **Descriptor** |
| … |  |
| for( n = 15; n >= 0; n− − ) { |  |
| xC = ( xS << 2 ) + ScanOrder[ 2 ][ scanIdx ][ n ][ 0 ] |  |
| yC = ( yS << 2 ) + ScanOrder[ 2 ][ scanIdx ][ n ][ 1 ] |  |
| if( sig\_coeff\_flag[ xC ][ yC ] ) { |  |
| if( numGreater1Flag < 8 ) { |  |
| **coeff\_abs\_level\_greater1\_flag**[ n ] | ae(v) |
| numGreater1Flag++ |  |
| if( coeff\_abs\_level\_greater1\_flag[ n ] && lastGreater1ScanPos = = −1 ) |  |
| lastGreater1ScanPos = n |  |
| } |  |
| if( lastSigScanPos = = −1 ) |  |
| lastSigScanPos = n |  |
| firstSigScanPos = n |  |
| } |  |
| } |  |
| signHidden = ( lastSigScanPos − firstSigScanPos > 3 && !cu\_transquant\_bypass\_flag ) |  |
| if ( cu\_transform\_skip\_flag && ( CuPredMode[ x0 ][ y0 ] == MODE\_INTRA ) && ( predModeIntra == INTRA\_DC || predModeIntra == INTRA\_PLANAR ) |  |
| signHidden = 0 |  |
| if( lastGreater1ScanPos != −1 ) |  |
| … |  |

**8.4.4.2.1 General intra sample prediction**

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 predModeIntra specifying the intra prediction mode,

– a variable nTbS specifying the transform block size,

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

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

* the residual sample array values r[ x ][ y ], with x, y = 0..nTbS−1.

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

The intra sample prediction process according to predModeIntra and trans\_quant\_bypass\_flag applies as follows:

* If predModeIntra is equal to INTRA\_PLANAR and trans\_quant\_bypass\_flag is equal to 1, the process specified in subclause 8.4.4.2.7 is applied.
* Otherwise the following process is applied:

The nTbS \* 4 + 1 neighbouring samples p[ x ][ y ] that are constructed samples prior to the deblocking filter process, with x = −1, y = −1..nTbS \* 2 − 1 and x = 0..nTbS \* 2 − 1, y = −1, are derived as follows:

– The neighbouring location (xNbCmp, yNbCmp ) is specified by:

( xNbCmp, yNbCmp ) = ( xTbCmp + x, yTbCmp + y ) (8‑27)

* The current luma location ( xTbY, yTbY ) and the neighbouring luma location (xNbY, yNbY ) are derived as follows:

( xTbY, yTbY ) = ( cIdx  = =  0 ) ? ( xTbCmp, yTbCmp ) : ( xTbCmp \* SubWidthC, yTbCmp \* SubHeightC ) (8‑28)

( xNbY, yNbY ) = ( cIdx  = =  0 ) ? ( xNbCmp, yNbCmp ) : ( xNbCmp \* SubWidthC, yNbCmp \* SubHeightC ) (8‑29)

* The availability derivation process for a block in z-scan order as specified in subclause 6.4.1 is invoked with the current luma location ( xCurr, yCurr ) set equal to ( xTbY, yTbY ) and the neighbouring luma location ( xNbY, yNbY ) as inputs, and the output is assigned to availableN.

– Each sample p[ x ][ y ] is derived as follows:

* If one or more of the following conditions are true, the sample p[ x ][ y ] is marked as "not available for intra prediction":
  + The variable availableN is equal to FALSE.
  + CuPredMode[ xNbY ][ yNbY ] is not equal to MODE\_INTRA and constrained\_intra\_pred\_flag is equal to 1.
* Otherwise, the sample p[ x ][ y ] is marked as "available for intra prediction" and the sample at the location ( xNbCmp, yNbCmp ) is assigned to p[ x ][ y ].

When at least one sample p[ x ][ y ] with x = −1, y = −1..nTbS \* 2 − 1 and x = 0..nTbS \* 2 − 1, y = −1 is marked as "not available for intra prediction", the reference sample substitution process for intra sample prediction in subclause 8.4.4.2.2 is invoked with the samples p[ x ][ y ] with x = −1, y = −1..nTbS \* 2 − 1 and x = 0..nTbS \* 2 − 1, y = −1, nTbS, and cIdx as inputs, and the modified samples p[ x ][ y ] with x = −1, y = −1..nTbS \* 2 − 1 and x = 0..nTbS \* 2 − 1, y = −1 as output.

Depending on the value of predModeIntra, the following ordered steps apply:

1. When cIdx is equal to 0 or ChromaArrayType is equal to 3, the filtering process of neighbouring samples specified in subclause 8.4.4.2.3 is invoked with the sample array p and the transform block size nTbS as inputs, and the output is reassigned to the sample array p.
2. The intra sample prediction process according to predModeIntra applies as follows:
   * If predModeIntra is equal to INTRA\_PLANAR, the corresponding intra prediction mode specified in subclause 8.4.4.2.4 is invoked with the sample array p and the transform block size nTbS as inputs, and the output is the predicted sample array predSamples.
   * Otherwise, if predModeIntra is equal to INTRA\_DC, the corresponding intra prediction mode specified in subclause 8.4.4.2.5 is invoked with the sample array p, the transform block size nTbS, and the colour component index cIdx as inputs, and the output is the predicted sample array predSamples.
   * Otherwise (predModeIntra is in the range of INTRA\_ANGULAR2..INTRA\_ANGULAR34), the corresponding intra prediction mode specified in subclause 8.4.4.2.6 is invoked with the intra prediction mode predModeIntra, the sample array p, the transform block size nTbS, and the colour component index cIdx as inputs, and the output is the predicted sample array predSamples.

**8.4.4.2.5 Specification of intra prediction mode INTRA\_DC**

Inputs to this process are:

– the neighbouring samples p[ x ][ y ], with x = −1, y = −1..nTbS \* 2 − 1 and x = 0..nTbS \* 2 − 1, y = −1,

– a variable nTbS specifying the transform block size,

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

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

If cu\_transquant\_bypass\_flag is equal to 0 and cu\_transform\_skip\_flag is equal to 0, the ~~The~~ values of the prediction samples predSamples[ x ][ y ], with x, y = 0..nTbS − 1, are derived by the following ordered steps:

1. A variable dcVal is derived as follows:

dcVal =  (8‑41)

where k = Log2( nTbS ).

1. Depending on the value of the colour component index cIdx, the following applies:

* If cIdx is equal to 0 and nTbS is less than 32, the following applies:

predSamples[ 0 ][ 0 ] = ( p[ −1 ][ 0 ] + 2 \* dcVal + p[ 0 ][ −1 ] + 2 )  >>  2 (8‑42)

predSamples[ x ][ 0 ] = ( p[ x ][ −1 ] + 3 \* dcVal + 2 )  >>  2, with x = 1..nTbS − 1 (8‑43)

predSamples[ 0 ][ y ] = ( p[ −1 ][ y ] + 3 \* dcVal + 2 )  >>  2, with y = 1..nTbS − 1 (8‑44)

predSamples[ x ][ y ] = dcVal, with x, y = 1..nTbS − 1 (8‑45)

* Otherwise, the prediction samples predSamples[ x ][ y ] are derived as follows:

predSamples[ x ][ y ] = dcVal, with x, y = 0..nTbS − 1 (8‑46)

Otherwise (if cu\_transquant\_bypass\_flag is equal to 1 or cu\_transform\_skip\_flag is equal to 1) the values of the prediction samples predSamples[ x ][ y ], with x, y = 0..nTbS − 1, are derived as follows:

– predSamples[x][y] = .

The neighboring samples p[ x ][ y ], with x, y = 0..nTbS − 1, are reconstructed by

– p[x][y] = r[x][y] + predSamples[x][y].

Add section 8.4.4.2.7

8.4.4.2.7 Specification of intra prediction mode INTRA\_SWP

The section describes the process which is invoked for prediction if trans\_quant\_bypass\_flag is equal to 1 and predModeIntra is equal to INTRA\_PLANAR.

Inputs to this process are:

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

– a variable nTbS specifying the transform block size,,

* the residual sample array values r[ x ][ y ], with x, y = 0..nTbS − 1.

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

The variable bitDepth is derived as follows:

– If cIdx is equal to 0, bitDepth is set equal to BitDepthY.

– Otherwise, bitDepth is set equal to BitDepthC.

SWP\_Table is set to {8, 6, 4, 3, 2, 2, 1, 1, 1} and SWP\_Max\_Argument is set to 8.

The nTbS \* 4 + 4 neighbouring samples p[ x ][ y ] that are constructed samples, with x = −2..−1, y = −2..nTbS−1 and x = −2..nTbS−1, y = −2..−1, are derived as follows:

– The neighbouring location (xNbCmp, yNbCmp ) is specified by ( xNbCmp, yNbCmp ) = ( xTbCmp + x, yTbCmp + y )

* The current luma location ( xTbY, yTbY ) and the neighbouring luma location (xNbY, yNbY) are derived as follows:
* ( xTbY, yTbY ) = ( cIdx  = =  0 ) ? ( xTbCmp, yTbCmp ) : ( xTbCmp \* SubWidthC, yTbCmp \* SubHeightC )
* ( xNbY, yNbY ) = ( cIdx  = =  0 ) ? ( xNbCmp, yNbCmp ) : ( xNbCmp \* SubWidthC, yNbCmp \* SubHeightC )
* The availability derivation process for a block in z-scan order as specified in subclause 6.4.1 is invoked with the current luma location ( xCurr, yCurr ) set equal to ( xTbY, yTbY ) and the neighbouring luma location ( xNbY, yNbY ) as inputs, and the output is assigned to availableN.
* Each sample p[ x ][ y ] is derived as follows:
* If the variable availableN is set to FALSE, p[ x ][ y ] is set to ( 1 << ( bitDepth – 1 ) ).
* Otherwise, the sample at the location ( xNbCmp, yNbCmp ) is assigned to p[ x ][ y ].

Each sample predSample[ x ][ y ] is derived as follows:

* SADopt is set to (1 << (bitDepth + 1))
* If x is smaller than ( nTbS – 1 ), candidate positions are given by N0 = {(m,n) | (0,-1), (1,-1), (-1,0), (-1,-1)}
* Otherwise, candidate positions are given by N0 = {(m,n) | (0,-1), (-1,0), (-1,-1)}
* For each entry from N0 the following process is applied:
  + SADm,n = | p[ x–1 ][ y ] – p[ x–1+m ][ y+n ] | + | p[ x ][ y–1 ] – p[ x+m ][ y–1+n ] |
  + If SADm,n  is smaller than SADopt, SADopt is set to SADm,n and popt is set to p[ x+m ][ y+n ]
  + If ( SADm,n >> ( bitDepth – 6 ) ) is greater than SWP\_Max\_Argument, wm,n is set to 0
  + Otherwise wm,n is set to SWP\_Table[ SADm,n >> ( bitDepth – 6 ) ]
* If SADopt is equal to zero, predSample[ x ][ y ] is set to popt
* Otherwise, predSample[ x ][ y ] is calculated depending on x and popt as follows:
  + If x is smaller than ( nTbS – 1 ):

predSample[ x ][ y ] = (p[ x–1 ][ y ] \* w-1,0 + p[ x–1 ][ y–1 ] \* w-1,-1 + p[ x ][ y–1 ] \* w0,-1

+ p[ x+1 ][ y–1 ] \* w1,-1 + popt\*(32 – w-1,0 –w-1,-1 –w0,-1 –w1,-1) + 16 ) >> 5.

* + Otherwise:

predSample[ x ][ y ] = (p[ x–1 ][ y ] \* w-1,0 + p[ x–1 ][ y–1 ] \* w-1,-1 + p[ x ][ y–1 ] \* w0,-1

+ popt\*(32 – w-1,0 –w-1,-1 –w0,-1) + 16 ) >> 5

* The neighbouring sample p[ x ][ y ] is updated as follows:

p[ x ][ y ] = r[ x ][ y ] + predSample[ x ][ y ]