### Residual coding syntax

|  |  |
| --- | --- |
| residual\_coding( x0, y0, log2TrafoWidth, log2TrafoHeight, scanIdx, cIdx ) { | Descriptor |
| **last\_significant\_coeff\_x\_prefix** | ae(v) |
| **last\_significant\_coeff\_y\_prefix** | ae(v) |
| if( last\_significant\_coeff\_x\_prefix > 3 ) |  |
| **last\_significant\_coeff\_x\_suffix** | ae(v) |
| if( last\_significant\_coeff\_y\_prefix > 3 ) |  |
| **last\_significant\_coeff\_y\_suffix** | ae(v) |
| numCoeff = 0 |  |
| do { |  |
| xC = ScanOrder[ log2TrafoWidth ][ log2TrafoHeight ][ scanIdx ][ numCoeff ][ 0 ] |  |
| yC = ScanOrder[ log2TrafoWidth ][ log2TrafoHeight ][ scanIdx ][ numCoeff ][ 1 ] |  |
| numCoeff++ |  |
| } while( ( xC != LastSignificantCoeffX ) | | ( yC != LastSignificantCoeffY ) ) |  |
| numLastSubset = (numCoeff − 1) >> 4 |  |
| for( i = numLastSubset; i >= 0; i− − ) { |  |
| offset = i << 4 |  |
| if( scanIdx = = 1 && log2TrafoWidth = = 3 && log2TrafoHeight = = 3 ) { |  |
| xCG = 0 |  |
| yCG = i |  |
| } else if( scanIdx = = 2 && log2TrafoWidth = = 3 && log2TrafoHeight = = 3 ) { |  |
| xCG = i |  |
| yCG = 0 |  |
| } else { |  |
| xCG = ScanOrder[ log2TrafoWidth − 2 ][ log2TrafoHeight − 2 ][ scanIdx ][ i ][ 0 ] |  |
| yCG = ScanOrder[ log2TrafoWidth − 2 ][ log2TrafoHeight − 2 ][ scanIdx ][ i ][ 1 ] |  |
| } |  |
| implicitNonZeroCoeff = 0 |  |
| if( (i < numLastSubset) && (i > 0) ) { |  |
| **significant\_coeff\_group\_flag**[ xCG ][ yCG ] | ae(v) |
| implicitNonZeroCoeff = 1 |  |
| } |  |
| for( n = 15; n >= 0; n− − ) { |  |
| xC = ScanOrder[ log2TrafoWidth ][ log2TrafoHeight ][ scanIdx ][ n + offset ][ 0 ] |  |
| yC = ScanOrder[ log2TrafoWidth ][ log2TrafoHeight ][ scanIdx ][ n + offset ][ 1 ] |  |
| if( (n + offset) < (numCoeff − 1) && significant\_coeff\_group\_flag[ xCG ][ yCG ] &&   ( n > 0 | | implicitNonZeroCoeff = = 0 ) ) { |  |
| **significant\_coeff\_flag**[ xC ][ yC ] | ae(v) |
| if( significant\_coeff\_flag[ xC ][ yC ] = = 1 ) |  |
| implicitNonZeroCoeff = 0 |  |
| } |  |
| } |  |
| firstNZPosInCG = 16 |  |
| lastNZPosInCG = −1 |  |
| numSigCoeff = 0 |  |
| firstGreater1CoeffIdx = −1 |  |
| for( n = 15; n >= 0; n− − ) { |  |
| xC = ScanOrder[ log2TrafoWidth ][ log2TrafoHeight ][ scanIdx ][ n + offset ][ 0 ] |  |
| yC = ScanOrder[ log2TrafoWidth ][ log2TrafoHeight ][ scanIdx ][ n + offset ][ 1 ] |  |
| if( significant\_coeff\_flag[ xC ][ yC ] ) { |  |
| if( numSigCoeff < 8 ) { |  |
| **coeff\_abs\_level\_greater1\_flag[** n **]** | ae(v) |
| numSigCoeff++ |  |
| if( coeff\_abs\_level\_greater1\_flag[ n ] && firstGreater1CoeffIdx = = −1 ) |  |
| firstGreater1CoeffIdx = n |  |
| } |  |
| if( lastNZPosInCG = = −1) |  |
| lastNZPosInCG = n |  |
| firstNZPosInCG = n |  |
| } |  |
| } |  |
| if (qpprime\_y\_zero\_transquant\_bypass\_flag ==1 && QP’Y ==0) |  |
| signHidden=0 |  |
| else |  |
| signHidden = ( lastNZPosInCG – firstNZPosInCG >= sign\_hiding\_threshold) ? 1 : 0 |  |
| if( firstGreater1CoeffIdx != −1 ) |  |
| **coeff\_abs\_level\_greater2\_flag[** firstGreater1CoeffIdx**]** | ae(v) |
| for( n = 15; n >= 0; n− − ) { |  |
| xC = ScanOrder[ log2TrafoWidth ][ log2TrafoHeight ][ scanIdx ][ n + offset ][ 0 ] |  |
| yC = ScanOrder[ log2TrafoWidth ][ log2TrafoHeight ][ scanIdx ][ n + offset ][ 1 ] |  |
| if( significant\_coeff\_flag[ xC ][ yC ] &&  (!sign\_data\_hiding\_flag | | !signHidden | | n != firstNZPosInCG) ) |  |
| **coeff\_sign\_flag[** n **]** | ae(v) |
| } |  |
| numSigCoeff = 0 |  |
| sumAbs = 0 |  |
| for( n = 15; n >= 0; n− − ) { |  |
| xC = ScanOrder[ log2TrafoWidth ][ log2TrafoHeight ][ scanIdx ][ n + offset ][ 0 ] |  |
| yC = ScanOrder[ log2TrafoWidth ][ log2TrafoHeight ][ scanIdx ][ n + offset ][ 1 ] |  |
| if( significant\_coeff\_flag[ xC ][ yC ] ) { |  |
| baseLevel = 1 + coeff\_abs\_level\_greater1\_flag[ n ] + coeff\_abs\_level\_greater2\_flag[ n ] |  |
| if( baseLevel = = ( ( numSigCoeff < 8 ) ? ( (n = = firstGreater1CoeffIdx) ? 3 : 2 ) : 1 ) ) |  |
| **coeff\_abs\_level\_remaining[** n **]** | ae(v) |
| transCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ] =   ( coeff\_abs\_level\_remaining[ n ] + baseLevel ) \* ( 1 − 2 \* coeff\_sign\_flag[ n ] ) |  |
| if( sign\_data\_hiding\_flag && signHidden ) { |  |
| sumAbs += ( coeff\_abs\_level\_remaining[ n ] + baseLevel ) |  |
| if( n = = firstNZPosInCG && (sumAbs%2 = = 1) ) |  |
| transCoeffLevel[x0][y0][cIdx][xC][yC] = −  transCoeffLevel[x0][y0][cIdx][xC][yC] |  |
| } |  |
| numSigCoeff++ |  |
| } else |  |
| transCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ] = 0 |  |
| } |  |
| } |  |
| } |  |



#### Sequence parameter set RBSP semantics

**profile\_idc** and **level\_idc** indicate the profile and level to which the coded video sequence conforms.

**reserved\_zero\_8bits** shall be equal to 0. Decoders shall ignore the value of reserved\_zero\_8bits.

**seq\_parameter\_set\_id** identifies the sequence parameter set that is referred to by the picture parameter set. The value of seq\_parameter\_set\_id shall be in the range of 0 to 31, inclusive.

**chroma\_format\_idc** specifies the chroma sampling relative to the luma sampling as specified in subclause . The value of chroma\_format\_idc shall be in the range of 0 to 3, inclusive. When chroma\_format\_idc is not present, it is inferred to be equal to 1 (4:2:0 chroma format).

**separate\_colour\_plane\_flag** equal to 1 specifies that the three colour components of the 4:4:4 chroma format are coded separately. separate\_colour\_plane\_flag equal to 0 specifies that the colour components are not coded separately. When separate\_colour\_plane\_flag is not present, it is inferred to be equal to 0. When separate\_colour\_plane\_flag is equal to 1, the coded picture consists of three separate components, each of which consists of coded samples of one colour plane (Y, Cb or Cr) that each use the monochrome coding syntax. In this case, each colour plane is associated with a specific colour\_plane\_id value.

NOTE 4 – There is no dependency in decoding processes between the colour planes having different colour\_plane\_id values. For example, the decoding process of a monochrome picture with one value of colour\_plane\_id does not use any data from monochrome pictures having different values of colour\_plane\_id for inter prediction.

Depending on the value of separate\_colour\_plane\_flag, the value of the variable ChromaArrayType is assigned as follows:

– If separate\_colour\_plane\_flag is equal to 0, ChromaArrayType is set equal to chroma\_format\_idc.

– Otherwise (separate\_colour\_plane\_flag is equal to 1), ChromaArrayType is set equal to 0.

**max\_temporal\_layers\_minus1** + 1 specifies the maximum number of temporal layers present in the sequence. The value of max\_temporal\_layers\_minus1 shall be in the range of 0 to 7, inclusive.

**pic\_width\_in\_luma\_samples** specifies the width of each decoded picture in units of luma samples. pic\_width\_in\_luma\_samples shall not be equal to 0 and shall be an integer multiple of ( 1 << Log2MinCbSize ).

**pic\_height\_in\_luma\_samples** specifies the height of each decoded picture in units of luma samples. pic\_height\_in\_luma\_samples shall not be equal to 0 and shall be an integer multiple of ( 1 << Log2MinCbSize ).

**pic\_cropping\_flag** equal to 1 indicates that the picture cropping offset parameters follow next in the sequence parameter set. pic\_cropping\_flag equal to 0 indicates that the picture cropping offset parameters are not present.

**pic\_crop\_left\_offset**, **pic\_crop\_right\_offset**, **pic\_crop\_top\_offset**, and **pic\_crop\_bottom\_offset** specify the samples of the pictures in the coded video sequence that are output from the decoding process, in terms of a rectangular region specified in picture coordinates for output.

The variables CropUnitX and CropUnitY are derived as follows:

– If ChromaArrayType is equal to 0, CropUnitX and CropUnitY are derived as:

CropUnitX = 1 (7-2)  
CropUnitY = 1 (7-3)

– Otherwise (ChromaArrayType is equal to 1, 2, or 3), CropUnitX and CropUnitY are derived as:

CropUnitX = SubWidthC (7-4)  
CropUnitY = SubHeightC (7-5)

The picture cropping rectangle contains luma samples with horizontal picture coordinates from CropUnitX \* pic\_crop\_left\_offset to PicWidthInSamplesL − ( CropUnitX \* pic\_crop\_right\_offset + 1 ) and vertical picture coordinates from CropUnitY \* pic\_crop\_top\_offset to PicHeightInSamplesL − ( CropUnitY \* pic\_crop\_bottom\_offset + 1 ), inclusive. The value of pic\_crop\_left\_offset shall be in the range of 0 to ( PicWidthInSamplesL / CropUnitX ) − ( pic\_crop\_right\_offset + 1 ), inclusive; and the value of pic\_crop\_top\_offset shall be in the range of 0 to ( PicHeightInSamplesL / CropUnitY ) − ( pic\_crop\_bottom\_offset + 1 ), inclusive.

When pic\_cropping\_flag is equal to 0, the values of pic\_crop\_left\_offset, pic\_crop\_right\_offset, pic\_crop\_top\_offset, and pic\_crop\_bottom\_offset is inferred to be equal to 0.

When ChromaArrayType is not equal to 0, the corresponding specified samples of the two chroma arrays are the samples having picture coordinates ( x / SubWidthC, y / SubHeightC ), where ( x, y ) are the picture coordinates of the specified luma samples.

NOTE – The picture cropping parameters are only applied at the output. All internal decoding processes are applied to the uncropped picture size.

**bit\_depth\_luma\_minus8** + 8 specifies the bit depth of the samples of the luma array and the value of the luma quantization parameter range offset QpBdOffsetY, as specified by

BitDepthY = 8 + bit\_depth\_luma\_minus8 (7‑6)  
QpBdOffsetY = 6 \* bit\_depth\_luma\_minus8 (7‑7)

bit\_depth\_luma\_minus8 shall be in the range of 0 to 6, inclusive.

**bit\_depth\_chroma\_minus8** + 8specifies the bit depth of the samples of the chroma arrays and the value of the chroma quantization parameter range offset QpBdOffsetC, as specified by

BitDepthC = 8 + bit\_depth\_chroma\_minus8 (7‑8)  
QpBdOffsetC = 6 \* bit\_depth\_chroma\_minus8 (7‑9)

bit\_depth\_chroma\_minus8 shall be in the range of 0 to 6, inclusive.

**pcm\_enabled\_flag** equal to 0 specifies that PCM data shall not be present in the video sequence.

**pcm\_sample\_bit\_depth\_luma\_minus1** + 1 specifies the number of bits used to represent each of PCM sample values of luma component. The value of pcm\_sample\_bit\_depth\_luma\_minus1 + 1 shall be less than or equal to the value of BitDepthY.

PCMBitDepthY = 1 + pcm\_sample\_bit\_depth\_luma\_minus1 (7‑10)

**pcm\_sample\_bit\_depth\_chroma\_minus1** + 1 specifies the number of bits used to represent each of PCM sample values of chroma components. The value of pcm\_sample\_bit\_depth\_chroma\_minus1 + 1 shall be less than or equal to the value of BitDepthC.

PCMBitDepthC = 1 + pcm\_sample\_bit\_depth\_chroma\_minus1 (7‑11)

**qpprime\_y\_zero\_transquant\_bypass\_flag** equal to 1 specifies that, when QP′Y is equal to 0, a lossless coding process shall be applied. In lossless coding operation, the scaling and transform process as specified in subclause and the in-loop filter process as specified in subclause are bypassed, and the sign bit hiding in residual\_coding() is not applied.