**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 { |  |
| if( intra\_block\_copy\_enabled\_flag ) |  |
| **intra\_bc\_flag**[ x0 ][ y0 ] | ae(v) |
| if( slice\_type != I && !intra\_bc\_flag[ x0 ][ y0 ] ) |  |
| **pred\_mode\_flag** | ae(v) |
| ~~if( palette\_mode\_enabled\_flag && ChromaArrayType = = 3 &&   CuPredMode[ x0 ][ y0 ] = = MODE\_INTRA && !intra\_bc\_flag[ x0 ][ y0 ] )~~ |  |
| if( palette\_mode\_enabled\_flag && ChromaArrayType != 0 &&   CuPredMode[ x0 ][ y0 ] = = MODE\_INTRA && !intra\_bc\_flag[ x0 ][ y0 ] ) |  |
| **palette\_mode\_flag**[ x0 ][ y0 ] | ae(v) |
| if( palette\_mode\_flag[ x0 ][ y0 ] ) |  |
| palette\_coding( x0, y0, nCbS ) |  |
| else { |  |
| if( CuPredMode[ x0 ][ y0 ] != MODE\_INTRA | | intra\_bc\_flag[ x0 ][ y0 ] | |   log2CbSize = = MinCbLog2SizeY ) |  |
| **part\_mode** | ae(v) |
| if( CuPredMode[ x0 ][ y0 ] = = MODE\_INTRA && !intra\_bc\_flag[ x0 ][ y0 ] ) { |  |
| if( PartMode = = PART\_2Nx2N && pcm\_enabled\_flag &&   log2CbSize >= Log2MinIpcmCbSizeY &&  log2CbSize <= Log2MaxIpcmCbSizeY ) |  |
| **pcm\_flag**[ x0 ][ y0 ] | ae(v) |
| if( pcm\_flag[ x0 ][ y0 ] ) { |  |
| while( !byte\_aligned( ) ) |  |
| **pcm\_alignment\_zero\_bit** | f(1) |
| pcm\_sample( x0, y0, log2CbSize ) |  |
| } else { |  |
| pbOffset = ( PartMode = = PART\_NxN ) ? ( nCbS / 2 ) : nCbS |  |
| for( j = 0; j < nCbS; j = j + pbOffset ) |  |
| for( i = 0; i < nCbS; i = i + pbOffset ) |  |
| **prev\_intra\_luma\_pred\_flag**[ x0 + i ][ y0 + j ] | ae(v) |
| for( j = 0; j < nCbS; j = j + pbOffset ) |  |
| for( i = 0; i < nCbS; i = i + pbOffset ) |  |
| if( prev\_intra\_luma\_pred\_flag[ x0 + i ][ y0 + j ] ) |  |
| **mpm\_idx**[ x0 + i ][ y0 + j ] | ae(v) |
| else |  |
| **rem\_intra\_luma\_pred\_mode**[ x0 + i ][ y0 + j ] | ae(v) |
| if( ChromaArrayType = = 3 ) |  |
| for( j = 0; j < nCbS; j = j + pbOffset ) |  |
| for( i = 0; i < nCbS; i = i + pbOffset ) |  |
| **intra\_chroma\_pred\_mode**[ x0 + i ][ y0 + j ] | ae(v) |
| else if( ChromaArrayType != 0 ) |  |
| **intra\_chroma\_pred\_mode**[ x0 ][ y0 ] | ae(v) |
| } |  |
| } else { |  |
| if( PartMode = = PART\_2Nx2N ) |  |
| prediction\_unit( x0, y0, nCbS, nCbS ) |  |
| else if( PartMode = = PART\_2NxN ) { |  |
| prediction\_unit( x0, y0, nCbS, nCbS / 2 ) |  |
| prediction\_unit( x0, y0 + ( nCbS / 2 ), nCbS, nCbS / 2 ) |  |
| } else if( PartMode = = PART\_Nx2N ) { |  |
| prediction\_unit( x0, y0, nCbS / 2, nCbS ) |  |
| prediction\_unit( x0 + ( nCbS / 2 ), y0, nCbS / 2, nCbS ) |  |
| } else if( PartMode = = PART\_2NxnU ) { |  |
| prediction\_unit( x0, y0, nCbS, nCbS / 4 ) |  |
| prediction\_unit( x0, y0 + ( nCbS / 4 ), nCbS, nCbS \* 3 / 4 ) |  |
| } else if( PartMode = = PART\_2NxnD ) { |  |
| prediction\_unit( x0, y0, nCbS, nCbS \* 3 / 4 ) |  |
| prediction\_unit( x0, y0 + ( nCbS \* 3 / 4 ), nCbS, nCbS / 4 ) |  |
| } else if( PartMode = = PART\_nLx2N ) { |  |
| prediction\_unit( x0, y0, nCbS / 4, nCbS ) |  |
| prediction\_unit( x0 + ( nCbS / 4 ), y0, nCbS \* 3 / 4, nCbS ) |  |
| } else if( PartMode = = PART\_nRx2N ) { |  |
| prediction\_unit( x0, y0, nCbS \* 3 / 4, nCbS ) |  |
| prediction\_unit( x0 + ( nCbS \* 3 / 4 ), y0, nCbS / 4, nCbS ) |  |
| } else { /\* PART\_NxN \*/ |  |
| prediction\_unit( x0, y0, nCbS / 2, nCbS / 2 ) |  |
| prediction\_unit( x0 + ( nCbS / 2 ), y0, nCbS / 2, nCbS / 2 ) |  |
| prediction\_unit( x0, y0 + ( nCbS / 2 ), nCbS / 2, nCbS / 2 ) |  |
| prediction\_unit( x0 + ( nCbS / 2 ), y0 + ( nCbS / 2 ), nCbS / 2, nCbS / 2 ) |  |
| } |  |
| } |  |
| if( !pcm\_flag[ x0 ][ y0 ] ) { |  |
| if( ( CuPredMode[ x0 ][ y0 ] != MODE\_INTRA &&   !( PartMode = = PART\_2Nx2N && merge\_flag[ x0 ][ y0 ] ) ) | |   ( CuPredMode[ x0 ][ y0 ] = = MODE\_INTRA && intra\_bc\_flag[ x0 ][ y0 ] ) ) |  |
| **rqt\_root\_cbf** | ae(v) |
| if( rqt\_root\_cbf ) { |  |
| if( residual\_adaptive\_colour\_transform\_enabled\_flag &&   ( CuPredMode[ x0 ][ y0 ] = = MODE\_INTER | | intra\_bc\_flag[ x0 ][ y0 ] | |   ( PartMode = = PART\_2Nx2N &&   intra\_chroma\_pred\_mode[ x0 ][ y0 ] = = 4 ) | |   (intra\_chroma\_pred\_mode[ x0 ][ y0 ] = = 4 &&   intra\_chroma\_pred\_mode[ x0 + nCbS/2 ][ y0 ] = = 4 &&   intra\_chroma\_pred\_mode[ x0 ][ y0 + nCbS/2 ] = = 4 &&   intra\_chroma\_pred\_mode[ x0 + nCbS/2 ][ y0 + nCbS/2 ] = = 4 ) ) ) |  |
| **cu\_residual\_act\_flag** | ae(v) |
| MaxTrafoDepth = ( CuPredMode[ x0 ][ y0 ] = = MODE\_INTRA ?   ( max\_transform\_hierarchy\_depth\_intra + IntraSplitFlag ) :   max\_transform\_hierarchy\_depth\_inter ) |  |
| transform\_tree( x0, y0, x0, y0, log2CbSize, 0, 0 ) |  |
| } |  |
| } |  |
| } |  |
| } |  |
| } |  |

**Palette syntax**

|  |  |
| --- | --- |
| palette\_coding( x0, y0, nCbS ) { | **Descriptor** |
| **palette\_share\_flag**[ x0 ][ y0 ] | ae(v) |
| if( !palette\_share\_flag[ x0 ][ y0 ] ) { |  |
| ~~palettePredictionFinished = 0~~ |  |
| ~~paletteNumPredictedEntries = 0~~ |  |
| Part1PalettePredictionFinished = 0 |  |
| Part1PaletteNumPredictedEntries = 0 |  |
| for( i = 0; i < Part1PredictorPaletteSize  && ! Part1PalettePredictionFinished &&  Part1PaletteNumPredictedEntries < palette\_max\_size; i++ ) { |  |
| **~~palette\_predictor\_run~~** | ~~ae(v)~~ |
| **part1\_palette\_predictor\_run** | ae(v) |
| if( part1\_palette\_predictor\_run != 1 ) { |  |
| if(part1\_palette\_predictor\_run > 1 ) |  |
| i += part1\_palette\_predictor\_run − 1 |  |
| Part1PalettePredictorEntryReuseFlag[ i ] = 1 |  |
| Part1PaletteNumPredictedEntries ++ |  |
| } else |  |
| Part1PalettePredictionFinished = 1 |  |
| } |  |
| if(ChromaArrayType = = 1 || ChromaArrayType = = 2){ |  |
| Part2PalettePredictionFinished = 0 |  |
| Part2PaletteNumPredictedEntries = 0 |  |
| for( i = 0; i < Part2PredictorPaletteSize  && ! Part2PalettePredictionFinished &&  (Part1PaletteNumPredictedEntries+ Part2PaletteNumPredictedEntries) < palette\_max\_size; i++ ) { |  |
| **part2\_palette\_predictor\_run** | ae(v) |
| if( part2\_palette\_predictor\_run != 1 ) { |  |
| if(part2\_palette\_predictor\_run > 1 ) |  |
| i += part2\_palette\_predictor\_run − 1 |  |
| Part2PalettePredictorEntryReuseFlag[ i ] = 1 |  |
| Part2PaletteNumPredictedEntries ++ |  |
| } else |  |
| Part2PalettePredictionFinished = 1 |  |
| } |  |
| } |  |
| paletteNumPredictedEntries = Part1PaletteNumPredictedEntries+ Part2PaletteNumPredictedEntries |  |
| if( paletteNumPredictedEntries < palette\_max\_size ) |  |
| **~~palette\_num\_signalled\_entries~~** | ~~ae(v)~~ |
| **part1\_palette\_num\_signalled\_entries** | ae(v) |
| If(ChromaArrayType = = 0) |  |
| compNum = 1 | ae(v) |
| else |  |
| compNum = 3 | ae(v) |
| for( cIdx = 0; cIdx < compNum; cIdx++ ) |  |
| ~~for( i = 0; i < palette\_num\_signalled\_ entries; i++ )~~ |  |
| **~~palette\_entry~~** | ~~ae(v)~~ |
| for( i = 0; i < part1\_palette\_num\_signalled\_ entries; i++ ) |  |
| **part1\_palette\_entry** | ae(v) |
| If((ChromaArrayType = = 1 || ChromaArrayType = = 2)&& (paletteNumPredictedEntries+ part1\_palette\_num\_signalled\_entries)< palette\_max\_size ){ |  |
| **part2\_palette\_num\_signalled\_entries** | ae(v) |
| for( i = 0; i < part2\_palette\_num\_signalled\_ entries; i++ ) |  |
| **part2\_palette\_entry** | ae(v) |
| } |  |
| } |  |
| if( currentPaletteSize != 0 ) |  |
| **palette\_escape\_val\_present\_flag** | ae(v) |
| if( palette\_escape\_val\_present\_flag ) { |  |
| if( cu\_qp\_delta\_enabled\_flag && !IsCuQpDeltaCoded ) { |  |
| **cu\_qp\_delta\_palette\_abs** | ae(v) |
| if( cu\_qp\_delta\_palette\_abs ) |  |
| **cu\_qp\_delta\_palette\_sign\_flag** | ae(v) |
| } |  |
| if( cu\_chroma\_qp\_offset\_enabled\_flag && !IsCuChromaQpOffsetCoded ) { |  |
| **cu\_chroma\_qp\_palette\_offset\_flag** | ae(v) |
| if( cu\_chroma\_qp\_offset\_flag && chroma\_qp\_offset\_list\_len\_minus1 > 0 ) |  |
| **cu\_chroma\_qp\_palette\_offset\_idx** | ae(v) |
| } |  |
| } |  |
| if( indexMax > 0) |  |
| **palette\_transpose\_flag** | ae(v) |
| scanPos = 0 |  |
| while( scanPos < nCbS \* nCbS ) { |  |
| xC = x0 + travScan[ scanPos ][ 0 ] |  |
| yC = y0 + travScan[ scanPos ][ 1 ] |  |
| if( scanPos > 0) { |  |
| xcPrev = x0 + travScan[ scanPos − 1 ][ 0 ] |  |
| ycPrev = y0 + travScan[ scanPos − 1 ][ 1 ] |  |
| } |  |
| if( indexMax > 0 && scanPos > = nCbS && palette\_run\_type\_flag[ xcPrev ][ ycPrev ]  ! = COPY\_ABOVE\_MODE ) { |  |
| **palette\_run\_type\_flag**[ xC ][ yC ] | ae(v) |
| } |  |
| if( palette\_run\_type\_flag[ xC ][ yC ] = = COPY\_INDEX\_MODE &&   adjustedIndexMax > 0) |  |
| **palette\_index\_idc** | ae(v) |
| if( indexMax > 0 ) { |  |
| maxPaletteRun = nCbS \* nCbS – scanPos – 1 |  |
| if( maxPaletteRun > 0 ) { |  |
| **palette\_run\_msb\_id\_plus1** | ae(v) |
| if( palette\_run\_msb\_id\_plus1 > 1 ) |  |
| **palette\_run\_refinement\_bits** | ae(v) |
| } |  |
| } else |  |
| paletteRun = nCbS \* nCbS − 1 |  |
| runPos = 0 |  |
| while ( runPos < = paletteRun ) { |  |
| xR = x0 + travScan[ scanPos ][ 0 ] |  |
| yR = y0 + travScan[ scanPos ][ 1 ] |  |
| if( palette\_run\_type\_flag[ xC ][ yC ] = = COPY\_ABOVE\_MODE) { |  |
| PaletteIndexMap[ xR ][ yR ] = PaletteIndexMap[ xR ][ yR − 1 ] |  |
| if(PaletteIndexMap[ xR ][ yR ] = = indexMax) |  |
| PaletteSampleMode[ xR ][ yR ] = ESCAPE\_MODE |  |
| else |  |
| PaletteSampleMode[ xR ][ yR ] = COPY\_ABOVE\_MODE |  |
| } |  |
| else{ |  |
| PaletteIndexMap[ xR ][ yR ] = paletteIndex |  |
| if(PaletteIndexMap[ xR ][ yR ] = = indexMax) |  |
| PaletteSampleMode[ xR ][ yR ] = ESCAPE\_MODE |  |
| else |  |
| PaletteSampleMode[ xR ][ yR ] = COPY\_INDEX\_MODE |  |
| } |  |
| ~~if( palette\_run\_type\_flag[ xC ][ yC ] = = COPY\_INDEX\_MODE &&   paletteIndex = = indexMax ) {~~ |  |
| if(PaletteSampleMode[ xR ][ yR ] = ESCAPE\_MODE){ |  |
| if( !ChromaSampleDecoded[  xR/ SubWidthC ][ yR/ SubHeightC  ] && (ChromaArrayType = = 1 || ChromaArrayType = = 2)) |  |
| **escape\_type** | ae(v) |
| else if (ChromaArrayType == 3) |  |
| escape\_type = 0 |  |
| else |  |
| escape\_type = 1 |  |
| if( escape\_type == 0) { |  |
| compNum = 3 |  |
| ChromaSampleDecoded[ xR/ SubWidthC ][ yR/ SubHeightC  ] = 1 |  |
| } |  |
| else |  |
| compNum = 1 |  |
| ~~for( cIdx = 0; cIdx < 3; cIdx++ ) {~~ |  |
| **~~palette\_escape\_val~~** | ~~ae(v)~~ |
| ~~PaletteEscapeVal[ cIdx ][ xR ][ yR ] = palette\_escape\_val~~ |  |
| for( cIdx = 0; cIdx < compNum; cIdx++ ) { |  |
| **palette\_escape\_val** | ae(v) |
| if(cIdx == 0) |  |
| PaletteEscapeVal[ cIdx ][ xR ][ yR ] = palette\_escape\_val |  |
| else |  |
| PaletteEscapeVal[ cIdx ][ xR/ SubWidthC ][ yR/ SubHeightC ] = palette\_escape\_val |  |
| } |  |
| ~~} else if(palette\_run\_type\_flag[ xC ][ yC ] = = COPY\_INDEX\_MODE ) {~~ |  |
| ~~PaletteSampleMode[ xR ][ yR ] = COPY\_INDEX\_MODE~~ |  |
| ~~PaletteIndexMap[ xR ][ yR ] = paletteIndex~~ |  |
| ~~} else {~~ |  |
| ~~PaletteSampleMode[ xR ][ yR ] = COPY\_ABOVE\_MODE~~ |  |
| ~~PaletteIndexMap[ xR ][ yR ] = PaletteIndexMap[ xR ][ yR − 1 ]~~ |  |
| } |  |
| runPos++ |  |
| scanPos++ |  |
| } |  |
| } |  |
| } |  |