**Palette Syntax**

|  |  |
| --- | --- |
| palette\_coding( x0, y0, nCbS ) { | Descriptor |
| palettePredictionFinished = 0 |  |
| NumPredictedPaletteEntries = 0 |  |
| for( i = 0; i < PredictorPaletteSize && !palettePredictionFinished &&  NumPredictedPaletteEntries < palette\_max\_size; i++ ) { |  |
| **palette\_predictor\_run** | ue(v) |
| if( palette\_predictor\_run != 1 ) { |  |
| if( palette\_predictor\_run > 1 ) |  |
| i += palette\_predictor\_run − 1 |  |
| PalettePredictorEntryReuseFlag[ i ] = 1 |  |
| NumPredictedPaletteEntries++ |  |
| } else |  |
| palettePredictionFinished = 1 |  |
| } |  |
| if( NumPredictedPaletteEntries < palette\_max\_size ) |  |
| **num\_signalled\_palette\_entries** | ue(v) |
| numComps = ( ChromaArrayType = = 0 ) ? 1 : 3 |  |
| for( cIdx = 0; cIdx < numComps; cIdx++ ) |  |
| for( i = 0; i < num\_signalled\_palette\_entries; i++ ) |  |
| **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( MaxPaletteIndex > 0) { |  |
| **palette\_transpose\_flag** | ae(v) |
| **num\_palette\_indices\_idc** | ae(v) |
| for( i=0; i < NumPaletteIndices; i++ ) { |  |
| **palette\_index\_idc** | ae(v) |
| PaletteIndexIdc[ i ] = palette\_index\_idc |  |
| } |  |
| **last\_palette\_run\_type\_flag** | ae(v) |
| } |  |
| CurrNumIndices = 0 |  |
| PaletteScanPos = 0 |  |
| while( PaletteScanPos < nCbS \* nCbS ) { |  |
| xC = x0 + travScan[ PaletteScanPos ][ 0 ] |  |
| yC = y0 + travScan[ PaletteScanPos ][ 1 ] |  |
| if( PaletteScanPos > 0) { |  |
| xcPrev = x0 + travScan[ PaletteScanPos − 1 ][ 0 ] |  |
| ycPrev = y0 + travScan[ PaletteScanPos − 1 ][ 1 ] |  |
| } |  |
| PaletteRun = nCbS \* nCbS − PaletteScanPos − 1 |  |
| if( MaxPaletteIndex > 0 && CurrNumIndices < NumPaletteIndices ) { |  |
| if( PaletteScanPos >= nCbS && palette\_run\_type\_flag[ xcPrev ][ ycPrev ]  != COPY\_ABOVE\_MODE && PaletteScanPos < nCbS \* nCbS – 1) { |  |
| **palette\_run\_type\_flag**[ xC ][ yC ] | ae(v) |
| } |  |
| readIndex = 0 |  |
| if( palette\_run\_type\_flag[ xC ][ yC ] = = COPY\_INDEX\_MODE &&   AdjustedMaxPaletteIndex > 0) |  |
| readIndex = 1 |  |
| maxPaletteRun = nCbS \* nCbS – PaletteScanPos – 1 |  |
| if( AdjustedMaxPaletteIndex > 0 &&   ( ( CurrNumIndices + readIndex ) < NumPaletteIndices | |   palette\_run\_type\_flag[ xC ][ yC ] != last\_palette\_run\_type\_flag ) ) |  |
| if( maxPaletteRun > 0 ) { |  |
| if (PaletteScanPos % nCbS == 0){ |  |
| **palette\_run\_end\_of\_line\_flag** | ae(v) |
| if (palette\_run\_end\_of\_line ){ |  |
| if (nCbS – 1 – PaletteScanPos/nCbs) |  |
| **palette\_num\_of\_rows\_minus1** | ae(v) |
| **}** |  |
| else{ |  |
| **palette\_run\_msb\_id\_plus1** | ae(v) |
| if( palette\_run\_msb\_id\_plus1 > 1 ) |  |
| **palette\_run\_refinement\_bits** | ae(v) |
| } |  |
| } |  |
| else{ |  |
| **palette\_run\_msb\_id\_plus1** | ae(v) |
| if( palette\_run\_msb\_id\_plus1 > 1 ) |  |
| **palette\_run\_refinement\_bits** | ae(v) |
| } |  |
| **~~palette\_run\_msb\_id\_plus1~~** | ~~ae(v)~~ |
| ~~if( palette\_run\_msb\_id\_plus1 > 1 )~~ |  |
| **~~palette\_run\_refinement\_bits~~** | ~~ae(v)~~ |
| } |  |
| CurrNumIndices + = readIndex |  |
| } |  |
| runPos = 0 |  |
| while ( runPos < = paletteRun ) { |  |
| xR = x0 + travScan[ PaletteScanPos ][ 0 ] |  |
| yR = y0 + travScan[ PaletteScanPos ][ 1 ] |  |
| if(palette\_run\_type\_flag[ xC ][ yC ] = = COPY\_INDEX\_MODE ) { |  |
| PaletteSampleMode[ xR ][ yR ] = COPY\_INDEX\_MODE |  |
| PaletteIndexMap[ xR ][ yR ] = CurrPaletteIndex |  |
| } else { |  |
| PaletteSampleMode[ xR ][ yR ] = COPY\_ABOVE\_MODE |  |
| PaletteIndexMap[ xR ][ yR ] = PaletteIndexMap[ xR ][ yR − 1 ] |  |
| } |  |
| runPos++ |  |
| PaletteScanPos++ |  |
| } |  |
| } |  |
| if( palette\_escape\_val\_present\_flag ) { |  |
| sPos = 0 |  |
| while( sPos < nCbS \* nCbS ) { |  |
| xC = x0 + travScan[ sPos ][ 0 ] |  |
| yC = y0 + travScan[ sPos ][ 1 ] |  |
| if( PaletteIndexMap[ xC ][ yC ] =  = MaxPaletteIndex ) { |  |
| for( cIdx = 0; cIdx < numComps; cIdx++ ) |  |
| if( cIdx = = 0 | |   ( xR % 2 = = 0 && yR % 2 = = 0 && ChromaArrayType = = 1 ) | |  ( xR % 2 = = 0 && ChromaArrayType = = 2 ) | |   ChromaArrayType = = 3 ) { |  |
| **palette\_escape\_val** | ae(v) |
| PaletteEscapeVal[ cIdx ][ xC ][ yC ] = palette\_escape\_val |  |
| } |  |
| } |  |
| sPos++ |  |
| } |  |
| } |  |
| } |  |

**Palette mode semantics**

The variable paletteRun specifies the number of consecutive locations minus 1 with the same palette index as the position in the above row when palette\_run\_type\_flag is equal to COPY\_ABOVE\_MODE or specifies the number of consecutive locations minus 1 with the same palette index when palette\_run\_type\_flag is equal to COPY\_INDEX\_MODE.

**palette\_end\_of\_line\_flag** equal to 1 specifies the run stops at the end of lines. palette\_end\_of\_line\_flag equal to 0 specifies the run does not stop at the end of a line.

**palette\_num\_of\_rows\_minus1** plus 1 specifies the number of consecutive rows where the run stops at the end of a line. When palette\_num\_of\_rows is not present, it is inferred to be equal to 0.

When (PaletteScanPos % nCbS) is equal to 0, paletteRun is derived as follows:

* When palette\_end\_of\_line\_flag is equal to 1,the following is applied:
  + A variable numPixel is set equal to palette\_num\_of\_rows\_minus1,
  + If PaletteScanPos is larger than 0, numPixel is refined as follows:

numPixel = (palette\_num\_of\_rows == 0) ? nCbS : (palette\_num\_of\_rows -1)

* paletteRun is set equal to min ((numPixel +1)\* nCbS -1, maxPaletteRun);
* Otherwise (palette\_end\_of\_line\_flag is equal to 0), the following is applied:
  + When palette\_run\_refinement\_bits is not present, paletteRun is inferred to be equal to 0.
  + When palette\_run\_msb\_id\_plus1 is present, the variable paletteRun is derived as follows:
  + If palette\_run\_msb\_id\_plus1 is greater than 1,

paletteRun = ( 1 << ( palette\_run\_msb\_id\_plus1 − 1 ) ) + palette\_run\_refinement\_bits (7 81)

* + Otherwise ( palette\_run\_msb\_id\_plus1 equal to to 0 or 1 ) paletteRun is set equal to palette\_run\_msb\_id\_plus1.

When (PaletteScanPos % nCbS) is not equal to 0, paletteRun is derived as follows:

* + A variable numRemainingLine is derived as follows:

numRemainingLine = nCbS – 1 – (PaletteScanPos % nCbS)

* + When palette\_run\_refinement\_bits is not present, paletteRun is inferred to be equal to 0.
  + When palette\_run\_msb\_id\_plus1 is present, the variable paletteRun is derived as follows:
  + If palette\_run\_msb\_id\_plus1 is greater than 1,
  + tempRun = ( 1 << ( palette\_run\_msb\_id\_plus1 − 1 ) ) + palette\_run\_refinement\_bits
  + Otherwise ( palette\_run\_msb\_id\_plus1 equal to to 0 or 1 ), paletteRun is set equal to palette\_run\_msb\_id\_plus1.
  + When paletteRun is larger than numRemainingLine, and paletteRun is not larger than (numRemainingLine + nCbS), and maxPaletteRun is not smaller than (numRemainingLine + nCbS), paletteRun is refined as follows:

A parameter RunPred is set equal to (numRemainingLine << 1) + 1;

A parameter Bound is set equal to *min*(nCbS - 1 - numRemainingLine, numRemainingLine);

paletteRun -= numRemainingLine + 1;

if (paletteRun <= (Bound << 1))

{

paletteRun = iRunPred + ((paletteRun & 1) ? -1 : 1) \* ((paletteRun + 1) >> 1);

}

else

{

if nCbS - 1 - numRemainingLine < numRemainingLine)

{

paletteRun = width - 1 - paletteRun;

}

paletteRun += numRemainingLine + 1;

}

**palette\_run\_msb\_id\_plus1** minus 1 specifies the index of the most significant bit in the binary representation of paletteRun. The value of palette\_run\_msb\_id\_plus1 shall be in the range of 0 to Floor( Log2( ( MaxPaletteRun ) ) + 1, inclusive.

**palette\_run\_refinement\_bits** specifies the refinement bits in the binary representation of paletteRun.

~~When palette\_run~~**~~\_~~**~~refinement\_bits is not present, it is inferred to be equal to 0.~~

~~When palette\_run\_msb\_id\_plus1 is present, the variable paletteRun is derived as follows:~~

* ~~If palette\_run~~**~~\_~~**~~msb\_id\_plus1 is greater than 1,~~

~~paletteRun = ( 1 << ( palette\_run\_msb\_id\_plus1 − 1 ) ) + palette\_run~~**~~\_~~**~~refinement\_bits (7‑81)~~

* ~~Otherwise ( palette\_run\_msb\_id\_plus1 equal to to 0 or 1 ) paletteRun is set equal to palette\_run\_msb\_id\_plus1.~~

9.3.3.1 General

|  |  |  |  |
| --- | --- | --- | --- |
| palette\_coding( ) | palette\_entries | FL | cMax = cIdx = = 0 ? ( (1<<BitDepthY) − 1 ) : ( (1<<BitDepthC) − 1 ) |
|  | palette\_escape\_val\_present\_flag | FL | cMax = 1 |
|  | cu\_qp\_delta\_palette\_abs | 9.3.3.11 | - |
|  | cu\_qp\_delta\_palette\_sign\_flag | FL | cMax = 1 |
|  | cu\_chroma\_qp\_palette\_offset\_flag | FL | cMax = 1 |
|  | cu\_chroma\_qp\_palette\_offset\_idx | TR | cMax = chroma\_qp\_offset\_list\_len\_minus1, cRiceParam = 0 |
|  | palette\_transpose\_flag | FL | cMax = 1 |
|  | palette\_run\_type\_flag | FL | cMax = 1 |
|  | num\_palette\_indices\_idc | 9.3.3.15 | nCbS, MaxPaletteIndex |
|  | last\_palette\_run\_type\_flag | FL | cMax = 1 |
|  | palette\_index\_idc | 9.3.3.14 | MaxPaletteIndex |
|  | palette\_run\_end\_of\_line\_flag | FL | cMax = 1 |
|  | palette\_num\_of\_rows\_minus1 | 9.3.3.16 | cRiceParam = 0 |
|  | palette\_run\_msb\_id\_plus1 | TR | cMax = Floor( Log2( MaxPaletteRun ) ) + 1, cRiceParam = 0 |
|  | palette\_run\_refinement\_bits | TB | cMax = ( ( 1<< palette\_run \_msb\_id\_plus1 ) >= MaxPaletteRun) ? MaxPaletteRun - ( 1 << ( palette\_run\_msb\_id\_plus1-1 ) ) : ( 1<< (palette\_run \_msb\_id\_plus1 – 1) ) – 1 |
|  | palette\_escape\_val | 9.3.3.13 | cIdx, qP |

#### 9.3.3.16 Binarization process for palette\_num\_of\_rows\_minus1

Input to this process is a request for a binarization for the syntax element palette\_num\_of\_rows\_minus1, and cRiceParam

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

The variable cMax is derived from cRiceParam as:

cMax = 4  <<  cRiceParam (9‑31)

The binarization of the syntax element palette\_num\_of\_rows\_minus1 is a concatenation of a prefix bin string and (when present) a suffix bin string.

For the derivation of the prefix bin string, the following applies:

* The prefix value of palette\_num\_of\_rows\_minus1, prefixVal, is derived as follows:

prefixVal = Min( cMax, palette\_num\_of\_rows\_minus1) (9‑32)

* The prefix bin string is specified by invoking the TR binarization process as specified in clause 9.3.3.2 for prefixVal with the variables cMax and cRiceParam as inputs.

When the prefix bin string is equal to the bit string of length 4 with all bits equal to 1, the suffix bin string is present and it is derived as follows:

* The suffix value of num\_palette\_indices\_idc, suffixVal, is derived as follows:

suffixVal = palette\_num\_of\_rows\_minus1− cMax (9‑33)

* The suffix bin string is specified by invoking the k-th order EGk binarization process as specified in clause 9.3.3.3 for the binarization of suffixVal with the Exp-Golomb order k set equal to cRiceParam + 1.

9.3.4.2.1 General

**Table 9‑47 – Assignment of ctxInc to syntax elements with context coded bins**

| **Syntax element** | **binIdx** | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **>= 5** |
| palette\_transpose\_flag | 0 | na | na | na | na | na |
| num\_palette\_indices\_idc | bypass | bypass | bypass | bypass | bypass | bypass |
| last\_palette\_run\_type\_flag | 0 | na | na | na | na | na |
| palette\_run\_type\_flag | 0 | na | Na | na | na | na |
| palette\_index\_idc | bypass | bypass | bypass | bypass | bypass | bypass |
| palette\_run\_end\_of\_line\_flag | 0 | na | na | na | na | na |
| palette\_num\_of\_rows\_minus1 | bypass | bypass | bypass | bypass | bypass | bypass |
| palette\_run\_msb\_id\_plus1 | (clause 9.3.4.2.8) | | | | | |
| palette\_run\_refinement\_bits | bypass | bypass | bypass | bypass | bypass | bypass |
| palette\_escape\_val | bypass | bypass | bypass | bypass | bypass | bypass |