#### 7.3.8.8 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) |
| **copy\_inter\_flag** | 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 |  |
| } |  |
| if(copy\_inter\_flag) |  |
| **last\_copy\_inter\_run\_type\_flag** | ae(v) |
| if(!last\_copy\_inter\_\_run\_type\_flag) |  |
| **last\_palette\_run\_type\_flag** | ae(v) |
| } |  |
| CurrNumIndices = 0 |  |
| PaletteScanPos = 0 |  |
| SignalRun = copy\_inter\_flag |  |
| 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(copy\_inter\_flag && !**copy\_inter\_run\_type\_flag**[ xcPrev ][ ycPrev ]) |  |
| **copy\_inter\_run\_type\_flag**[ xC ][ yC ] | ae(v) |
| if( MaxPaletteIndex > 0 && CurrNumIndices < NumPaletteIndices ) { |  |
| if( PaletteScanPos >= nCbS && palette\_run\_type\_flag[ xcPrev ][ ycPrev ]  != COPY\_ABOVE\_MODE && PaletteScanPos < nCbS \* nCbS – 1  && !copy\_inter\_flag) { |  |
| **palette\_run\_type\_flag**[ xC ][ yC ] | ae(v) |
| } |  |
| readIndex = 0 |  |
| if(palette\_run\_type\_flag [ xC ][ yC ] = = COPY\_INDEX\_MODE &&   AdjustedMaxPaletteIndex > 0) |  |
| readIndex = 1 |  |
| if(readIndex == 1 && (((palette\_run\_type\_flag[ xcPrev ][ ycPrev ] == COPY\_ABOVE\_MODE && PaletteIndexMap[ xC ][ yC-1 ]== COPY\_INTER\_INDEX) || palette\_run\_type\_flag[ xcPrev ][ ycPrev ] == COPY\_INTER\_MODE)  && CurrPaletteIndex == CurrentPaletteSize - 2) |  |
| **palette\_index\_refinement\_bits** |  |
| if(palette\_index\_refinement\_bits == 1) |  |
| CurrPaletteIndex = CurrentPaletteSize - 1 |  |
| maxPaletteRun = nCbS \* nCbS – PaletteScanPos – 1 |  |
| if(copy\_inter\_flag && ( CurrNumIndices+ readIndex >= NumPaletteIndices && RunType[ xC ][ yC ] == LastRunType){ |  |
| **last\_run\_flag** | ae(v) |
| SignalRun = !last\_run\_flag |  |
| } |  |
| if(SignalRun || ( AdjustedMaxPaletteIndex > 0 &&   ( ( CurrNumIndices + readIndex ) < NumPaletteIndices | |   palette\_run\_type\_flag[ xC ][ yC ] != last\_palette\_run\_type\_flag ) ) |  |
| if( maxPaletteRun > 0) { |  |
| **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 if(palette\_run\_type\_flag[ xC ][ yC ] = = COPY\_ABOVE\_MODE) { |  |
| PaletteSampleMode[ xR ][ yR ] = COPY\_ABOVE\_MODE |  |
| PaletteIndexMap[ xR ][ yR ] = PaletteIndexMap[ xR ][ yR − 1 ] |  |
| }else { |  |
| PaletteSampleMode[ xR ][ yR ] = COPY\_INTER\_MODE |  |
| PaletteIndexMap[ xR ][ yR ] = -2 |  |
| } |  |
| } |  |
| 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++ |  |
| } |  |
| } |  |
| if(copy\_inter\_flag) |  |
| prediction\_unit( x0, y0, nCbS, nCbS ) |  |
| } |  |

**7.4.9.6** **Palette mode semantics**

**copy\_inter\_flag** equal to 1 specifies that the current coding unit is coded by inter palette mode. palette\_mode\_flag equal to 0 specifies that the current coding unit is not coded by inter palette mode.

When copy\_inter\_flag is not present, it is inferred to be equal to 0.

**last\_copy\_inter\_run\_type\_flag** equal to 1 specifies that the last one run type is COPY\_INTER mode.

A variable LastRunType is set equal to last\_copy\_inter\_run\_type\_flag? COPY\_INTER\_MODE : (last\_palette\_run\_type\_flag ? COPY\_ABOVE\_MODE : COPY\_INDEX\_MODE )

**copy\_inter\_run\_type\_flag** equal to 1 specifies that the current palette index is equal to the palette index at the same location inside the index map of the block indicated by the decoded block vector.

A variable RunType[ xC ][ yC ] is set equal to copy\_inter\_run\_type\_flag[ xC ][ yC ] ? COPY\_INTER\_MODE : (palette\_run\_type\_flag[ xC ][ yC ] ? COPY\_ABOVE\_MODE : COPY\_INDEX\_MODE )

**palette\_index\_refinement\_bits** equal to 1 specifies that the current palette index is equal to CurrentPaletteSize minus 1

**last\_run\_flag** equal to 1 specifies that the current run is the last run in the current block.