**Draft Text specification for SCCE4 Test 3.4**

This text specification is on top of JCTVC-Q0034\_v4/JCTVC-Q0176\_v2.

## Syntax

**7.3.2.2 Sequence parameter set RBSP synt****ax**

|  |  |
| --- | --- |
| seq\_parameter\_set\_rbsp( ) { | Descriptor |
| **sps\_video\_parameter\_set\_id** | u(4) |
| **sps\_max\_sub\_layers\_minus1** | u(3) |
| … |  |
| … |  |
| **amp\_enabled\_flag** | u(1) |
| **sample\_adaptive\_offset\_enabled\_flag** | u(1) |
| **sps\_hybrid\_color\_pixel\_search\_eabled\_flag** | u(1) |
| **pcm\_enabled\_flag** | u(1) |
| …. |  |
| … |  |
| } |  |

**7.3.8.5 Coding unit syntax**

|  |  |
| --- | --- |
| coding\_unit( x0, y0, log2CbSize ) { | Descriptor |
| … |  |
| if( cu\_skip\_flag[ x0 ][ y0 ] ) |  |
| prediction\_unit( x0, y0, nCbS, nCbS ) |  |
| else { |  |
| **color\_pixel\_search\_flag** | ae(v) |
| if( color\_pixel\_search\_flag ) { |  |
| **single\_color\_mode** | ae(v) |
| nCbS = 1 << log2CbSize |  |
| if( single\_color\_mode ) |  |
| for( ch = 0; ch < 3; ch++ ) |  |
| **single\_color\_value** | u(8) |
| else |  |
| color\_pixel\_search\_coding(x0, y0, nCbS, nCbS) |  |
| } |  |
| } else { |  |
| … |  |
| } |  |
| 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 ] && color\_pixel\_search\_flag[ x0 ][ y0 ] ) |  |
| **rqt\_root\_cbf** | ae(v) |
| if( rqt\_root\_cbf ) { |  |
| … |  |
| } |  |
| } |  |
| } |  |
| } |  |

|  |  |
| --- | --- |
| color\_pixel\_search\_coding(x0, y0, nCbS, nCbS ) { | Descriptor |
| **color\_pixel\_search\_scan\_dir** | ae(v) |
| if( sps\_hybrid\_color\_pixel\_search\_eabled\_flag && nCbS = = MAX\_CU\_SIZE ) |  |
| hybrid\_color\_pixel\_search\_eabled\_flag = 1 |  |
| else |  |
| hybrid\_color\_pixel\_search\_eabled\_flag = 0 |  |
| for( i = 0; i < nCbS \* nCbS ) { |  |
| **unmatch\_found\_flag** | ae(v) |
| if( unmatch\_found\_flag ) { |  |
| **adptive\_idx\_found\_flag** | ae(v) |
| if(adptive\_idx\_found\_flag) { |  |
| **adptive\_idx** | ae(v) |
| } else { |  |
| for( ch = 0; ch < 3; ch++ ) |  |
| **color\_value** | u(8) |
| } |  |
| } else { |  |
| if( hybrid\_color\_pixel\_search\_eabled\_flag ) |  |
| **2d\_block\_search\_flag** | ae(v) |
| if( 2d\_block\_search\_flag ) { |  |
| **2d\_distance** | ae(v) |
| **2d\_length** | ae(v) |
| } else { |  |
| **sig\_dist\_flag** | ae(v) |
| if( sig\_dist\_flag ) |  |
| **dist\_equal\_one\_flag** | ae(v) |
| else |  |
| **distance\_minus2** | ae(v) |
| **sig**\_**len\_ flag** | ae(v) |
| if ( sig\_len\_ flag ) **{** |  |
| **length\_minus1** | ae(v) |
| i += length\_minus1 + 1; |  |
| } else { |  |
| **length\_minus\_cuWidth** | ae(v) |
| i += length\_minus\_cuWidth + cuWidth; |  |
| } |  |
| } |  |
| } |  |
| } |  |
| } |  |

7.4.3.2 Sequence parameter set RBSP semantics

**sps\_hybrid\_color\_pixel\_search\_eabled\_flag** equal to 1 specifies that hybrid\_color\_pixel\_search\_eabled\_flag is present in the SPS RBSP syntax structure. sps\_hybrid\_color\_pixel\_search\_eabled\_flag equal to 0 specifies hybrid\_color\_pixel\_search\_eabled\_flag is not present.

## Semantics

**color\_pixel\_search\_flag[**x0][y0]: equal to 1 specifies that the color pixel search mode is selected.; **color\_pixel\_search\_flag**[x0][y0] equal to 0 specifies that the color pixel search mode is not selected;

**single\_color\_mode[x0][y0]:** equal to 1 specifies that current CU has only 1 entry in its color table. **single\_color\_mode[x0][y0]** equal to 0 specifies that current CU has more than 1 entry in its color table.

**single\_color\_value[x0][y0][n][0]:** specifies the value of the first/only color entry of current color table.

**color\_pixel\_search\_scan\_dir[**x0][y0]: equal to 1 specifies that the index map of current coding unit uses the horizontal scanning; color\_pixel\_search\_scan\_dir[x0][y0] equal to 0 specifies that the index map of current coding unit uses the vertical scanning;

**hybrid\_color\_pixel\_search\_eabled\_flag**: equal to 1 specifies that current coding unit is using hybrid 1D and 2D search mode. **hybrid\_color\_pixel\_search\_eabled\_flag** equal to 0 specifies that current coding unit uses only 1D search method.

**unmatch\_found\_flag[x0][y0][cIdx]:** equal to 1 specifies that current index could not find its match from reference buffer; **unmatch\_found\_flag[x0][y0][cIdx]:** equal to 0 specifies that current index could find its match from reference buffer;

**adptive\_idx\_found\_flag[x0][y0][cIdx]:** equal to 1 specifies that the pixel value represented by adptive\_idx is found in adaptive color table for unmatched data. **adptive\_idx\_found\_flag[x0][y0][cIdx]:** equal to 0 specifies that the pixel value represented by adptive\_idx is not found in adaptive color table for unmatched data.

**adptive\_idx[x0][y0][n][adptIdx]:** specifies the index found in adaptive color table.

**color\_value[x0][y0][n][cIdx]:** specifies the pixel value represented by adptive\_idx.

The adaptive color table for unmatched data generation process is described by following pseudo-code.

if(adp\_idx\_found\_flag) {

decodeValue(adptive\_idx);

} else {

decodeValue(color\_value[0]);

decodeValue(color\_value[1]);

decodeValue(color\_value[2]);

for(adptive\_idx = 0; adptive\_idx < uiDictMaxSizeSofar; ++ adptive\_idx)

if( (color\_value[0] == adptPal[0][adptive\_idx]) &&

(color\_value[1] == adptPal[1][adptive\_idx]) &&

(color\_value[2] == adptPal[2][adptive\_idx]) )

break;

if(adptive\_idx == uiDictMaxSizeSofar ) {

adptPal[0][adptive\_idx] = color\_value[0];

adptPal[1][adptive\_idx] = color\_value[1];

adptPal[2][adptive\_idx] = color\_value[2];

++uiDictMaxSizeSofar;

}

}

**2d\_block\_search\_flag[x0][y0]:** equal to 1 specifies that the type of current matched pair is a 2D block match. **2d\_block\_search\_flag[x0][y0]** equal to 0 specifies that the type of current matched pair is a 1D string match.

**2d\_distance[x0][y0][cIdx]:** specifies the distance of current matched pair. It is the raster scan index of top-left corner of the matched reference block within the reference search window.

**2d\_length[x0][y0][cIdx]:** specifies the length of current matched 2D pair.

The variable 2D width and height are derived as follows:

2DWidth**[x0][y0][cIdx]** = (**2d\_length[x0][y0][cIdx]**%(1+minw-MIN\_2D\_WIDTH))+MIN\_2D\_WIDTH;

2DHeight**[x0][y0][cIdx]** = (**2d\_length[x0][y0][cIdx]**/(1+minw-MIN\_2D\_WIDTH))+MIN\_2D\_HEIGHT;

**sig\_dist\_flag[x0][y0][cIdx]:** equal to 1 specifies that the distance of current matched pair is significant distance; sig\_dist\_flag[x0][y0][cIdx] equal to 0 specifies that the distance of current matched pair is not significant distance;

**dist\_equal\_one\_flag[x0][y0][cIdx]:** equal 1 specifies that the distance equals to 1; dist\_equal\_one\_flag[x0][y0][cIdx]equal 0 specifies that the distance equals to the width of current coding unit.

**distance\_minus2[x0][y0][cIdx]:** specifies the distance minus 2 of current matched pair.

The variable distance is derived as follows:

distance = distance\_minus2[x0][y0][cIdx] + 2;

**sig\_len\_flag[x0][y0][cIdx]:** equal to 1 specifies that the length of current matched pair is significant length; sig\_len\_flag[x0][y0][cIdx] equal to 0 specifies that the length of current matched pair is not significant length;

**length\_minus1[x0][y0][cIdx]:** specifies the length minus 1 of current matched pair.

The variable length is derived as follows:

Length = length\_minus [x0][y0][cIdx] + 1;

cIdx = cIdx + length;

**length\_minus\_cuWidth[x0][y0][cIdx]::** specifies the length minus the width of current coding unit for current matched pair.

The variable length is derived as follows:

Length = length\_minus\_cuWidth[x0][y0][cIdx] + cuWidth;

cIdx = cIdx + length;