### Working Draft

### Derivation of prediction mode

### The text is modified as Table 1 when the prediction mode is derived for the pixel below ESCAPE index.

|  |
| --- |
| ... |
| if( scanPos > = nCbS && palette\_mode[xC\_prev][yC\_prev] ! = COPY\_ABOVE && palette\_mode[xC\_prev][yC\_prev-1] != ESCAPE) |
| **palette\_mode**[ xC ][ yC ] |
| … |

### Limiting number of run for ‘copy above’ mode

The text is modified as below when the number of run is limited by the pixel below ESCAPE index.

#### Palette mode semantics

**palette\_index** specifies an index to the array represented by palette\_entries. If palette\_index (after necessary adjustments) is equal to the value of the palette\_size, the sample is inferred to be coded in ESCAPE mode. [Ed. (YK/RJ): Check the need of adding dimension indices into syntax element name, and specify the inference of values in the semantics for those not explicitly signalled.] [Ed. (GJS): Check/fix usage of "necessary adjustments"]

When palette\_index is not present, it is inferred to be equal to 0.

The palette indices for samples that are not coded in ESCAPE mode are represented by the array paletteMap[ xC ][ yC ]. The array indices xC and yC specify the location ( xC, yC ) of the sample relative to the top-left luma sample of the picture.

If palette\_mode[ xC ][ yC ] is equal to ESCAPE, the variable IndexEscape and IndexBelowEscape are derived as follows :

* If palette\_transpose\_flag is equal to 0, the following applies :
  + If yC is even, IndexBelowEscape shall be ( yC )\*nCbs+ nCbs –xC and IndexBelowEscape shall be ( yC+1 )\*nCbs+ nCbs –xC.
  + If yC is odd, IndexBelowEscape shall be ( yC )\*nCbs+ xC and IndexBelowEscape shall be ( yC+1 )\* nCbs +xC
* If palette\_transpose\_flag is equal to 1, the following applies :
  + If xC is even, IndexBelowEscape shall be be ( xC )\* nCbs + nCbs -yC and IndexBelowEscape shall be ( xC+1 )\* nCbs + nCbs -yC
  + If xC is odd, IndexEscape shall be ( xC+1 )\* nCbs +yC and IndexBelowEscape shall be ( xC+1 )\* nCbs +yC

**palette\_run** the number of consecutive locations minus 1 with the same palette index as the position in the row aboe when palette\_mode is equal to COPY\_ABOVE or represents the number of consecutive locations minus 1 with the same palette index when the palette\_mode is equal to INDEX. When

The value of MaxPaletteRun is derived as follows:

If palette\_mode[ xC ][ yC ] is equal to COPY\_ABOVE, the variable IndexCurrent is derived as follows :

* If palette\_transpose\_flag is equal to 0, the following applies :
  + If yC is even, IndexCurrent shall be (yC)\*nCbs+ nCbs -xC
  + If yC is odd, IndexCurrent shall be (yC)\* nCbs +xC
* If palette\_transpose\_flag is equal to 1, the following applies :
  + If xC is even, IndexCurrent shall be (xC)\* nCbs + nCbs -yC
  + If xC is odd, IndexCurrent shall be (xC)\* nCbs +yC

If IndexCurrent is greater than IndexEscape and less than IndexBelowEscape, MaxPaletteRun shall be IndexBelowEscape – IndexCurrent. Otherwise MaxPaletteRun shall be same as nCbs \* nCbs.

#### 9.3.3.13 Binarization process for palette\_run

Input to this process is a request for a binarization for the syntax element palette\_run and colour component index cIdx.

Output of this process is the binarization of palette\_run as specified in if palette\_mode is equal to INDEX and the binarization of palette\_run as specified in Table 9-47 if palette\_mode is equal to COPY\_ABOVE.

Table 9‑46 – Binarization for palette\_run if palette\_mode is equal to

|  |  |
| --- | --- |
| **palette\_run** | Codeword |
| 0 | '0' |
| 1 | '10' |
| 2 | '110' |
| >2 | Prefix = '111', Suffix = TR code , cRiceParam = 3 |

Table 9‑47 – Binarization for palette\_run

|  |  |  |
| --- | --- | --- |
| **palette\_run** | Codeword |  |
|  | prefix | suffix |
| 0 | '0' |  |
| 1 | '10' |  |
| >1 | Prefix = TR code , cRiceParam = 1 | Truncated Binary 9.3.3.6 (cMax= MaxPaletteRun) |