|  |  |
| --- | --- |
| slice\_header( ) { | Descriptor |
| **first\_slice\_in\_pic\_flag** | u(1) |
| if( RapPicFlag ) |  |
| **no\_output\_of\_prior\_pics\_flag** | u(1) |
| **pic\_parameter\_set\_id** | ue(v) |
| if( !first\_slice\_in\_pic\_flag ) |  |
| **slice\_address** | u(v) |
| if( dependent\_slice\_enabled\_flag && !first\_slice\_in\_pic\_flag ) |  |
| **dependent\_slice\_flag** | u(1) |
| if( !dependent\_slice\_flag ) { |  |
| **slice\_type** | ue(v) |
| if( output\_flag\_present\_flag ) |  |
| **pic\_output\_flag** | u(1) |
| if( separate\_colour\_plane\_flag = = 1 ) |  |
| **colour\_plane\_id** | u(2) |
| if( !IdrPicFlag ) { |  |
| **pic\_order\_cnt\_lsb** | u(v) |
| **short\_term\_ref\_pic\_set\_sps\_flag** | u(1) |
| if( !short\_term\_ref\_pic\_set\_sps\_flag ) |  |
| short\_term\_ref\_pic\_set( num\_short\_term\_ref\_pic\_sets ) |  |
| else |  |
| **short\_term\_ref\_pic\_set\_idx** | u(v) |
| if( long\_term\_ref\_pics\_present\_flag ) { |  |
| if( num\_long\_term\_ref\_pics\_sps > 0 ) |  |
| **num\_long\_term\_sps** | ue(v) |
| **num\_long\_term\_pics** | ue(v) |
| for( i = 0; i < num\_long\_term\_sps + num\_long\_term\_pics; i++ ) { |  |
| if( i < num\_long\_term\_sps ) |  |
| **lt\_idx\_sps**[ i ] | u(v) |
| else { |  |
| **poc\_lsb\_lt**[ i ] | u(v) |
| **used\_by\_curr\_pic\_lt\_flag**[ i ] | u(1) |
| } |  |
| **delta\_poc\_msb\_present\_flag**[ i ] | u(1) |
| if( delta\_poc\_msb\_present\_flag[ i ] ) |  |
| **delta\_poc\_msb\_cycle\_lt**[ i ] | ue(v) |
| } |  |
| } |  |
| } |  |
| if( sample\_adaptive\_offset\_enabled\_flag ) { |  |
| **slice\_sao\_luma\_flag** | u(1) |
| **slice\_sao\_chroma\_flag** | u(1) |
| } |  |
| if( slice\_type = = P | | slice\_type = = B ) { |  |
| if( sps\_temporal\_mvp\_enable\_flag ) |  |
| **slice\_temporal\_mvp\_enable\_flag** | u(1) |
| **num\_ref\_idx\_active\_override\_flag** | u(1) |
| if( num\_ref\_idx\_active\_override\_flag ) { |  |
| **num\_ref\_idx\_l0\_active\_minus1** | ue(v) |
| if( slice\_type = = B ) |  |
| **num\_ref\_idx\_l1\_active\_minus1** | ue(v) |
| } |  |
| if( lists\_modification\_present\_flag ) |  |
| ref\_pic\_list\_modification( ) |  |
| if( slice\_type = = B ) |  |
| **mvd\_l1\_zero\_flag** | u(1) |
| if( cabac\_init\_present\_flag ) |  |
| **cabac\_init\_flag** | u(1) |
| if( slice\_temporal\_mvp\_enable\_flag ) { |  |
| if( slice\_type = = B ){ |  |
| **collocated\_from\_l0\_flag** | u(1) |
| **dv\_collocated\_from\_l0\_flag** | u(1) |
| } |  |
| if( ( collocated\_from\_l0\_flag && num\_ref\_idx\_l0\_active\_minus1 > 0 )  | | ( !collocated\_from\_l0\_flag &&  num\_ref\_idx\_l1\_active\_minus1 > 0 ) ){ |  |
| **collocated\_ref\_idx** | ue(v) |
| **dv\_collocated\_ref\_idx** | ue(v) |
| } |  |
| } |  |
| if( ( weighted\_pred\_flag && slice\_type = = P) | |  ( weighted\_bipred\_flag && slice\_type = = B ) ) |  |
| pred\_weight\_table( ) |  |
| **five\_minus\_max\_num\_merge\_cand** | ue(v) |
| } |  |
| **slice\_qp\_delta** | se(v) |
| if( pic\_slice\_level\_chroma\_qp\_offsets\_present\_flag ) { |  |
| **slice\_cb\_qp\_offset** | se(v) |
| **slice\_cr\_qp\_offset** | se(v) |
| } |  |
| if( deblocking\_filter\_control\_present\_flag ) { |  |
| if( deblocking\_filter\_override\_enabled\_flag ) |  |
| **deblocking\_filter\_override\_flag** | u(1) |
| if( deblocking\_filter\_override\_flag ) { |  |
| **slice\_header\_disable\_deblocking\_filter\_flag** | u(1) |
| if( !slice\_header\_disable\_deblocking\_filter\_flag ) { |  |
| **beta\_offset\_div2** | se(v) |
| **tc\_offset\_div2** | se(v) |
| } |  |
| } |  |
| } |  |
| if( loop\_filter\_across\_slices\_enabled\_flag &&  ( slice\_sao\_luma\_flag | | slice\_sao\_chroma\_flag | |   !disable\_deblocking\_filter\_flag ) ) |  |
| **slice\_loop\_filter\_across\_slices\_enabled\_flag** | u(1) |
| } |  |
| if( tiles\_enabled\_flag | | entropy\_coding\_sync\_enabled\_flag ) { |  |
| **num\_entry\_point\_offsets** | ue(v) |
| if( num\_entry\_point\_offsets > 0 ) { |  |
| **offset\_len\_minus1** | ue(v) |
| for( i = 0; i < num\_entry\_point\_offsets; i++ ) |  |
| **entry\_point\_offset**[ i ] | u(v) |
| } |  |
| } |  |
| if( slice\_header\_extension\_present\_flag ) { |  |
| **slice\_header\_extension\_length** | ue(v) |
| for( i = 0; i < slice\_header\_extension\_length; i++) |  |
| **slice\_header\_extension\_data\_byte**[ i ] | u(8) |
| } |  |
| byte\_alignment( ) |  |
| } |  |

G.8.5.2.1.16 Derivation process for a disparity vector from temporal neighbour blocks

Inputs to this process are

* a luma location ( xP, yP ) specifying the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,
* variables specifying the width and the height of the luma prediction block, nPbW and nPbH

Outputs of this process are

* the disparity vector mvDisp,
* the availability flag availableFlag.
* the flag availableFlagIvpMvTempX specifying whether the disparity vector used for inter-view motion vector prediction of the temporal collocated block is available.
* the disparity vector mvDispIvpTempX used for inter-view motion vector prediction of the temporal collocated block

The flag availableFlagIvpMvTempX and the components of mvDispIvpTempX are set equal to 0.

Depending on the values of slice\_type, **dv\_collocated\_from\_l0\_flag**, and **dv\_collocated\_ref\_idx**, the variable colPic, specifying the picture that contains the collocated partition, is derived as follows.

* If slice\_type is equal to B and **dv\_collocated\_from\_l0\_flag** is equal to 0, the variable colPic specifies the picture that contains the collocated partition as specified by RefPicList1[ **dv\_collocated\_ref\_idx** ].
* Otherwise (slice\_type is equal to B and **dv\_collocated\_from\_l0\_flag** is equal to 1 or slice\_type is equal to P), the variable colPic specifies the picture that contains the collocated partition as specified by RefPicList0[ **dv\_collocated\_ref\_idx** ].

The prediction block colPb and the luma location ( xPCol, yPCol ) specifing the top-left sample of the colPb are derived in the following ordered steps:

1. The vertical component of the right bottom luma location of the current luma prediction block is derived as follows.

yPRb = yP + nPbH (G‑)

* + If ( yP >> Log2CtbSizeY ) is equal to ( yPRb >> Log2CtbSizeY ), the horizontal component of the right-bottom luma location of the current luma prediction block is derived by

xPRb = xP + nPbW (G‑)

and the prediction block colPb is set as the luma prediction block covering the modified location given by ( ( xPRb >> 4 ) << 4, ( yPRb >> 4 ) << 4 ) inside the colPic.

* + Otherwise ( ( yP >> Log2CtbSizeY ) is not equal to ( yPRb >> Log2CtbSizeY ) ), colPb is marked as "unavailable".

1. When colPb is coded in an intra prediction mode or colPb is marked as "unavailable", the following applies.
   * Central luma location of the current prediction block is derived by

xPCtr = ( xP + ( nPbW >> 1 ) (G‑)

yPCtr = ( yP + ( nPbH >> 1 ) (G‑)

* + The variable colPb is set as the luma prediction block covering the modified location given by ( ( xPCtr >> 4 ) << 4, ( yPCtr >> 4 ) << 4 ) inside the colPic.

1. ( xPCol, yPCol ) is set equal to the top-left sample of the colPb relative to the top-left luma sample of the colPic.
2. The derivation process for a disparity vector in a block of a candidate picture as specified in subclause G.8.5.2.1.14 is invoked with candidate picture colPic, luma location ( xPCol, yPCol ) as inputs, and the flag availableFlag, the disparity vector mvDisp, the flag availableFlagIvpMvTempX, and a disparity vector mvDispIvpTempX as output.