#### Sequence parameter set RBSP syntax

|  |  |
| --- | --- |
| seq\_parameter\_set\_rbsp( ) { | Descriptor |
| **profile\_idc** | u(8) |
| **reserved\_zero\_8bits** /\* equal to 0 \***/** | u(8) |
| **level\_idc** | u(8) |
| **seq\_parameter\_set\_id** | ue(v) |
| **max\_temporal\_layers\_minus1** | u(3) |
| **pic\_width\_in\_luma\_samples** | u(16) |
| **pic\_height\_in\_luma\_samples** | u(16) |
| **bit\_depth\_luma\_minus8** | ue(v) |
| **bit\_depth\_chroma\_minus8** | ue(v) |
| **pcm\_bit\_depth\_luma\_minus1** | u(4) |
| **pcm\_bit\_depth\_chroma\_minus1** | u(4) |
| **log2\_max\_frame\_num\_minus4** | ue(v) |
| **pic\_order\_cnt\_type** | ue(v) |
| if( pic\_order\_cnt\_type = = 0 ) |  |
| **log2\_max\_pic\_order\_cnt\_lsb\_minus4** | ue(v) |
| else if( pic\_order\_cnt\_type = = 1 ) { |  |
| **delta\_pic\_order\_always\_zero\_flag** | u(1) |
| **offset\_for\_non\_ref\_pic** | se(v) |
| **num\_ref\_frames\_in\_pic\_order\_cnt\_cycle** | ue(v) |
| for( i = 0; i < num\_ref\_frames\_in\_pic\_order\_cnt\_cycle; i++ ) |  |
| **offset\_for\_ref\_frame[ i ]** | se(v) |
| } |  |
| **max\_num\_ref\_frames** | ue(v) |
| **gaps\_in\_frame\_num\_value\_allowed\_flag** | u(1) |
| **log2\_min\_coding\_block\_size\_minus3** | ue(v) |
| **log2\_diff\_max\_min\_coding\_block\_size** | ue(v) |
| **log2\_min\_transform\_block\_size\_minus2** | ue(v) |
| **log2\_diff\_max\_min\_transform\_block\_size** | ue(v) |
| **log2\_min\_pcm\_coding\_block\_size\_minus3** | ue(v) |
| **max\_transform\_hierarchy\_depth\_inter** | ue(v) |
| **max\_transform\_hierarchy\_depth\_intra** | ue(v) |
| **chroma\_pred\_from\_luma\_enabled\_flag** | u(1) |
| **loop\_filter\_across\_slice\_flag** | u(1) |
| **sample\_adaptive\_offset\_enabled\_flag** | u(1) |
| **adaptive\_loop\_filter\_enabled\_flag** | u(1) |
| **pcm\_loop\_filter\_disable\_flag** | u(1) |
| **cu\_qp\_delta\_enabled\_flag** | u(1) |
| **temporal\_id\_nesting\_flag** | u(1) |
| **inter\_4x4\_enabled\_flag** | u(1) |
| **quater\_pixel\_threshold\_idx\_p\_slice** | u(2) |
| **quater\_pixel\_threshold\_idx\_b\_slice** | u(2) |
| rbsp\_trailing\_bits( ) |  |
| } |  |

#### **Sequence parameter set RBSP semantics**

**quater\_pixel\_threshold\_idx\_p** specifies the quarter pixel threshold value TH for P slice according to Table 7-2. The TH shall be used for the derivation process for luma motion vector prediction in subclause 8.4.2.1.7 and the derivation process for luma motion vector in subclause 8.4.2.1.11.

Table 7-2 Quarter pixel threshold value

|  |  |
| --- | --- |
| **quarter\_pixel\_threshold\_idx\_p/b** | Quarter pixel threshold value TH |
| 0 | TH=0 |
| 1 | TH=2 |
| 2 | TH=4 |
| 3 | TH=infinite |

**quater\_pixel\_threshold\_idx\_b** specifies the quarter pixel threshold value TH for B slice according to Table 7-2. The TH shall be used for the derivation process for luma motion vector prediction in subclause 8.4.2.1.7 and the derivation process for luma motion vector in subclause 8.4.2.1.11.

#### 8.4.2.1 Derivation process for motion vector components and reference indices

Input to this process are

* a luma location ( xC, yC ) of the top-left luma sample of the current coding unit relative to the top-left luma sample of the current picture,
* a luma location ( xB, yB ) of the top-left luma sample of the current prediction unit relative to the top-left luma sample of the current coding unit,
* variables specifying the width and the height of the prediction unit for luma, nPSW and nPSH,
* a variable PartIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are

* luma motion vectors mvL0 and mvL1 and chroma motion vectors mvCL0 and mvCL1,
* reference indices refIdxL0 and refIdxL1,
* prediction list utilization flags predFlagL0 and predFlagL1.

Let ( xP, yP ) specify the top-left luma sample of the current prediction unit relative to the top-left luma sample of the current picture where xP = xC + xB and yP = yC + yB.

For the derivation of the variables mvL0 and mvL1, refIdxL0 and refIdxL1 as well as PredFlagL0 and PredFlagL1, the following applies.

* If PredMode is equal to MODE\_SKIP, the derivation process for luma motion vectors for merge mode as specified in subclause 8.4.2.1.3 is invoked with the luma location ( xP, yP ), variables nPSW, nPSH and the partition index PartIdx as inputs and the output being the luma motion vectors mvL0, mvL1, the reference indices refIdxL0, refIdxL1, and the prediction list utilization flags predFlagL0 and predFlagL1.
* Otherwise, if PredMode is equal to MODE\_INTER and merge\_flag[ xP ][ yP ] is equal to 1,, the derivation process for luma motion vectors for merge mode as specified in subclause 8.4.2.1.3 is invoked with the luma location ( xP, yP ), variables nPSW and nPSH and the partition index PartIdx as inputs and the outputs being the luma motion vectors mvL0 and mvL1, the reference indices refIdxL0 and refIdxL1, the prediction utilization flags predFlagL0 and predFlagL1.
* Otherwise, for X being replaced by either 0 or 1 in the variables predFlagLX, mvLX, refIdxLX and in Pred\_LX and in the syntax elements ref\_idx\_lX and mvd\_lX, the following applies.

1. The variables LcToLx, refIdxLX and predFlagLX are derived as follows.

* If inter\_pred\_flag[ xP ][ yP ] is equal to Pred\_LC and PredLCToPredLx[ ref\_idx\_lc[ xP ][ yP ] ] is equal to Pred\_LX,

refIdxLX = RefIdxLCToRefIdxLx[ ref\_idx\_lc[ xP ][ yP ] ] (8‑52) predFlagLX = 1 (8‑53) mvd\_lX[ xP ][ yP ][ 0 ] = mvd\_lc[ xP ][ yP ][ 0 ] (8‑54)  
 mvd\_lX[ xP ][ yP ][ 1 ] = mvd\_lc[ xP ][ yP ][ 1 ] (8‑55)  
 mvp\_idx\_lX[ xP ][ yP ][ 0 ] = mvp\_idx\_lc[ xP ][ yP ] (8‑56)  
 LcToLx = LX (8‑57)

* Otherwise, if inter\_pred\_flag[ xP ][ yP ] is equal to Pred\_LX or Pred\_BI,

refIdxLX = ref\_idx\_lX[ xP ][ yP ] (8‑58)  
 predFlagLX = 1 (8‑59)

* Otherwise, the variables refIdxLX and predFlagLX are specified by

refIdxLX = -1 (8‑60)  
 predFlagLX = 0 (8‑61)

1. The variable mvdLX is derived as follows.

mvdLX[ 0 ] = mvd\_lX[ xP ][ yP ][ 0 ] (8‑62)  
 mvdLX[ 1 ] = mvd\_lX[ xP ][ yP ][ 1 ] (8‑63)

1. When predFlagLX is equal to 1, the variable mvpLX is derived as follows.

* The derivation process for luma motion vector prediction in subclause 8.4.2.1.5 is invoked with the luma location ( xP, yP ), variables nPSW and nPSH and refIdxLX as the inputs and the output being mvpLX.

1. When predFlagLX is equal to 1, the derivation process for luma motion vector in subclause 8.4.2.1.11 is invoked with the variables mvpLX and mvdLX as the inputs and the output being the luma motion vector mvLX.

When ChromaArrayType is not equal to 0 and predFlagLX (with X being either 0 or 1) is equal to 1, the derivation process for chroma motion vectors in subclause 8.4.2.1.8 is invoked with mvLX and refIdxLX as inputs and the output being mvCLX.

##### Derivation process for luma motion vector prediction

Inputs to this process are

* a luma location ( xP, yP ) specifying the top-left luma sample of the current prediction unit relative to the top-left sample of the current picture,
* variables specifying the width and the height of the prediction unit for luma, nPSW and nPSH.
* the reference index of the current prediction unit partition refIdxLX (with X being 0 or 1).

Output of this process is

* the prediction mvpLX of the motion vector mvLX (with X being 0 or 1).

The motion vector predictor mvpLX is derived in the following ordered steps.

1. The derivation process for motion vector predictor candidates from neighboring prediction unit partitions in subclause is invoked with luma location ( xP, yP ), the width and the height of the prediction unit nPSW and nPSH, and refIdxLX (with X being 0 or 1, respectively) as inputs and the availability flags availableFlagLXN and the motion vectors mvLXN with N being replaced by A, B as the output.
2. The derivation process for temporal luma motion vector prediction in subclause is invoked with luma location ( xP, yP ) , the width and the height of the prediction unit nPSW and nPSH, and refIdxLX (with X being 0 or 1, respectively) as the inputs and with the output being the availability flag availableFlagLXCol and the temporal motion vector predictor mvLXCol.
3. The, is constructed of which elements are given as specified order:
4. mvLXA, if availableFlagLXA is equal to 1
5. mvLXB, if availableFlagLXB is equal to 1
6. mvLXCol, if availableFlagLXCol is equal to 1
7. The variable numMVPCandLX is set to the number of elements within the mvpListLX. If numMVPCandLX is greater than 0, the following applies:

If TH, which is derived from Table 7-2, is equal to 0, the following applies:

For mvpListLX[i] with i=0, …, numMVPCandLX-1, the following applies:

mvpListLX[i][0] = mvpListLX[i][0]>>1<<1

mvpListLX[i][1] = mvpListLX[i][1]>>1<<1

1. When motion vectors have the same value, the motion vectors are removed from the list except the motion vector which has the smallest order in the mvpListLX.
2. When mvpListLX is empty, a zero motion vector is added as follows.

mvpListLC[ 0 ][0] = 0 (8‑128)

mvpListLC[ 0 ][1] = 0 (8‑129)

1. The variable numMVPCandLX is set to the number of elements within the mvpListLX and maxNumMVPCand is set to 2.
2. The motion vector predictor list is modifed to contain exactly maxNumMVPCand motion vector predictor candidates as follows.
   * + If numMVPCandLX is less than maxNumMVPCand, the derivation process for zero motion vector predictor candidates specified in subclause is invoked with mvpListLX and numMVPCandLX given as input and the output is assigned to mvpListLX and numMVPCandLX.
     + Otherwise (numMVPCandLX is equal to or greater than maxNumMVPCand), all motion vector predictor candidates mvpListLX[ idx ] with idx greater than maxNumMVPCand − 1 are removed from the list.
3. The motion vector of mvpListLX[ mvp\_idx\_lX[ xP, yP ] ] is assigned to mvpLX.

##### Derivation process for luma motion vector

Inputs to this process are

a luma motion vector prediction mvpLX

a luma motion vector difference mvdLX

Outputs to this process is

The luma motion vector mvLX

The variables CTR is derived as follows.

CTR[0] = (mvpLX [0]>>1<<1) – mvpLX[0]

CTR[1] = (mvpLX [1]>>1<<1) – mvpLX[1]

The motion vector mvLX is derived as follows.

If Abs(mvdLX[0]-CTR[0])>TH

mvLX[0] = mvpLX[0] + mvdLX[0]\*2-CTRq[0]-Sign(mvdLX[0]-CTRq[0])\*TH

mvLX[1] = mvpLX[1] + mvdLX[1]\*2+CTRq[1]

Otherwise if Abs(mvdLX[1]-CTR[1])>TH

mvLX[0] = mvpLX[0] + mvdLX[0]\*2+CTRq[0]

mvLX[1] = mvpLX[1] + mvdLX[1]\*2-CTRq[1]-Sign(mvdLX[1]-CTRq[1])\*TH

otherwise

mvLX[0] = mvpLX[0] + mvdLX[0]

mvLX[1] = mvpLX[1] + mvdLX[1]

##### Derivation process for chroma motion vectors

[Ed.: (WJ) 4:2:0 assumption yet]

Inputs to this process are a luma motion vector mvLX and a reference index refIdLX.

Output of this process is a chroma motion vector mvCLX.

A chroma motion vector is derived from the corresponding luma motion vector.

For the derivation of the chroma motion vector mvCLX, the following applies.

mvCLX[ 0 ] = mvLX[ 0 ] (8‑153)

mvCLX[ 1 ] = mvLX[ 1 ] (8‑154)