|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11**  10th Meeting: Stockholm, Sweden, July 11-20, 2012 | Document: JCTVC-J0083  M25405 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **AHG9: On slice header parsing overhead reduction** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Minhua Zhou Texas Instruments Inc., USA  Mihir Mody Texas Instruments Inc., India | Tel: Email:  Tel  Email: | +1-214-480-3816 [zhou@ti.com](mailto:zhou@ti.com)  +91-80-2509 9307  mihir@ti.zom |
| *Source:* | Texas Instruments Inc; | | |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Abstract

In the current HM7.0 design there is a huge difference (over 100x) in slice header parsing overhead between the typical and worst case. The worst case slice header parsing can be beyond capability of real-time decoding in which slice header parsing is normally implemented in software. Weighted prediction tables and reference picture parameter sets (RPS) are parsing intensive parts of slice header. To reduce the worst case slice header parsing overhead, this contribution proposes the following two changes: 1) move RPS from slice header to APS, 2) reduce weighted prediction table cycle overhead by either constraining the total number of reference pictures (option 1), or enabling signaling of default weighted prediction tables in APS and limiting the number of slice headers per picture which can override weighted prediction tables signaled in APS (option 2). The proposed methods can reduce the worst case slice parsing overhead by roughly 2x to 3x based on cycle estimate on ARM9.

# Introduction

Slice header contains high-level parameters that are needed for decoding the LCU data of a slice, those parameters include picture order counter, reference picture parameter sets, SAO (sample adaptive offset) control parameters, reference picture list parameters and modification parameters, de-blocking filter control parameters, weighted prediction tables, in-loop filter slice on/off control flag and sub-stream entries, etc.

In the current HEVC design (JCTVC-I1003) design, the parsing intensive parameters defined in the slice header include the short and long term reference picture parameter sets (RPS), and the weighted prediction tables. Although carrying those parameters in the slice header is optional, in the worst (evil) case a picture can contain a maximum number of slices permitted for the profile and level, and have all the slices in the picture carry those parsing intensive parameters all the time.

There is a significant difference in slice header parsing overhead between typical and worst case. As shown in Table 1, settings for number of slices per frame, weighted prediction tables, and reference picture lists can be quite different in the typical and worst case, which leads to a huge gap between the tow cases.

|  |  |  |
| --- | --- | --- |
|  | Typical case | Worst case |
| Number of slices per frame | 4 | 136 (e.g. in H.264/AVC) |
| Number of active reference frames (sets of weighted prediction tables) | Two sets of weights are used and hence number of weight values are 2 per direction. | Weights are present for all 16 entries per direction and hence 32 weights are present for B slice |
| Chroma weighted prediction tables | Weights are used for Luma only, since fade in / fade out scenarios will have changes of intensity and not change in coloring | Weights are present for Luma and Chroma. Further, in chroma different weights are present for Cb and Cr |
| Ref. pic list parameters | Not present | Present for every slice |

**Table 1. Example of typical vs. worst case settings for slice header parsing in H.264/AVC at 1080p@30**

Table 2 summarizes the AMR9 cycle estimate of slice header parsing for H.264/AVC and HEVC at 1080p@30. As shown in Table 2, there is an over 400x difference and 100x difference between the worst and typical case slice parsing overhead for AVC and HEVC, respectively. Although the worst case slice header parsing overhead of HEVC has been significantly reduced (roughly by 4x) when compared to H.264/AVC, thanks to the reduced maximum number of slices per picture and much simplified reference picture list re-ordering process, it is still too high for a real-time decoder to handle.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Slice header parsing ARM9 cycle estimate (1080p@30) | | | | |
| INPUT | **AVC** | | **HEVC (HM7.0)** | |
|  | Typical | Worst | Typical | Worst |
| DPB size (#frame) | 5 | 5 | 6 | 6 |
| Number of active reference frames | 2 | 16 | 2 | 16 |
| Chroma weights? (0->no, 1->yes) | 0 | 1 | 0 | 1 |
| Ref. Pic List (0-> no, 1->yes) | 0 | 1 | 0 | 1 |
| Slice type (1->P-slice, 2->B-slice) | 2 | 2 | 2 | 2 |
| Number of slices per frame | 4 | 136 | 4 | 65 |
| frame-rate (fps) | 30 | 30 | 30 | 30 |
| Number of weighted prediction table overrides per frame |  |  |  |  |
|  |  |  |  |  |
| OUTPUT (cycles) |  |  |  |  |
| Basic slice header parsing | 9000 | 9000 | 9500 | 9500 |
| Weighted prediction tables parsing | 1464 | 35136 | 1464 | 35136 |
| Ref pic list modification parsing | 0 | 9160 | 0 | 7460 |
| RPS parsing | 0 | 4420 | 0 | 4420 |
| Ref pic re-ordering (operation) | 0 | 54400 | 0 | 4800 |
| One slice header: total cycles | 10464 | 112116 | 10964 | 61316 |
| **Total MHz for slice header parsing and operation** | **1** | **457** | **1** | **120** |

**Table 2. Typical vs. worst case AM9 cycle estimate for slice header parsing in H.264/AVC and HEVC at 1080p@30**

In typical video decoder designs, slice header parsing is normally implemented in software for flexibility and error concealment reasons. Even if a 4x more powerful processor (than ARM9) is used for slice parsing in actual products, the worst case slice parsing overhead is still around 30 MHz for 1080p@30. For a 1080p@120 decoder at 300 MHz, the slice header parsing would account for about 40% overall cycle count for the decoder, which is way too high when considering that the typical margin for high-level syntax paring is only around 10% of overall decoder cycle budget.

# Proposed solution

To reduce the worst case slice parsing overhead, it is proposed to focus on the worst case cycle reduction for RPS and weighted prediction tables, which are the most parsing intensive parts of slice header.

## RPS slice header parsing cycle reduction

In the current HM7.0 design, the same RPS is repeated in every slice header, which is a significant overhead when there are many slices in a picture. Because RPS is used only once per picture, it is proposed to move the RPS from slice header to APS.

In the APS, aps\_ref\_pic\_set\_present\_flag is added to signal whether the short term and long term reference picture parameter sets (RPS) are present in the APS, and the relevant RPS syntax elements are moved from slice header to APS. aps\_ref\_pic\_set\_present\_flag is recommended to set to zero if the current picture referring to the APS is an instantaneous decoding refresh (IDR) picture.

## Weighted prediction tables slice header parsing cycle reduction

The parsing overhead of weighted prediction tables is dictated by number of active reference pictures. In the current design, num\_ref\_idx\_l0\_active\_minus1 and num\_ref\_idx\_l1\_active\_minus1 are both in the range of [0:15], meaning that in the worst case there can be 16 reference pictures in each direction. We propose the following two options for achieving the cycle reduction goal.

Option 1: the first option requires no syntax change, and achieves cycle reduction by reducing the worst case number of reference pictures. It is proposed that the sum of num\_ref\_idx\_l0\_active\_minus1 and num\_ref\_idx\_l1\_active\_minus1 is constrained to be less than or equal to 6, meaning that in the worst case there can only be a total of 8 reference pictures. Compared to the current design, this restriction provides a factor of 4 (i.e. (16+16)/8) cycle overhead reduction for parsing of weighted prediction tables.

Since the total number of reference pictures in the DBP is 5, this restriction still allows applying different weights for a same reference pictures for 3 out of 5 reference pictures.

Note that with the proposed restriction, num\_ref\_idx\_l0\_active\_minus1 for a P-slice is in the range of [0:7], inclusive. num\_ref\_idx\_l0\_active\_minus1 + num\_ref\_idx\_l1\_active\_minus1 for a B-slice is in the range of [0:6], inclusive. If num\_ref\_idx\_l0\_active\_minus1 and num\_ref\_idx\_l1\_active\_minus1 are inherited from the PPS, the same restriction applies.

Reducing the total number of reference pictures is also helpful for reducing parsing overhead of reference picture list modification.

Option 2: the second option still allows the maximum number of reference pictures up to 16 in each direction, but requires syntax changes. In the proposed method, the default weighted prediction tables are carried in the APS, and a limited number of weighted prediction table overrides is allowed in slice header to enable changing weights within a picture. The major parts of option 2 are

1. Weighted prediction tables are moved from the slice header to the adaptation parameter set (APS). In the APS, aps\_pred\_weight\_table\_present\_flag is added to the APS to signal whether the default weighted prediction tables are present in the APS. It is recommended to set aps\_pred\_weight\_table\_present\_flag to zero if the current picture referring to the APS does not contain any B- or P-slices.
2. Since there might be need of changing weighted prediction tables from slice to slice of a picture, a weighted prediction table override flag, pred\_weighted\_table\_override\_flag, is added to the slice header. If this flag is set, an explicit weighted prediction table is transmitted in the slice header and used for the current slice. Otherwise, the weighted prediction table transmitted in the APS is used for the slice.
3. To limit parsing overhead of weighted prediction tables in a picture, the number of slices in the picture that can carry their own weighted prediction tables are constrained. It is constrained that up to N slices in a picture can carry their own weighted prediction tables, while the rest of slices in the picture have to use the default weighted prediction table carried in the APS, where N is 1 for level 3.1 and below, and 4 for level 4 and above.

# Cycle Estimate

Table 3 summarizes the ARM9 cycle estimate for slice header parsing with the proposed methods at 1080p@30. The major observations are.

1. Moving RPS from slice header to APS can reduce overhead by roughly 8 MHz
2. Option 2 can reduce more overhead than option 1 by roughly 13 MHz
3. Overall the worst case slice header can be cut by 2x to 3x with proposed methods.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Slice header parsing ARM9 cycle estimate (1080p@30) | | | | | | | |
| INPUT | **HEVC (HM7.0)** | | **HEVC (proposed)** | | | | |
|  | Typical | Worst | Typical | Worst (option1 + RPS in slice header) | Worst (option1 + RPS in APS) | Worst (option2 + RPS in slice header) | Worst (option2 + RPS in APS) |
| DPB size (#frame) | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| Number of active reference frames | 2 | 16 | 2 | 4 | 4 | 16 | 16 |
| Chroma weights? (0->no, 1->yes) | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| Ref. Pic List (0-> no, 1->yes) | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| Slice type (1->P-slice, 2->B-slice) | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| Number of slices per frame | 4 | 65 | 4 | 65 | 65 | 65 | 65 |
| frame-rate (fps) | 30 | 30 | 30 | 30 | 30 | 30 | 30 |
| Number of weighted prediction table overrides per frame |  |  |  |  |  | 4 | 4 |
|  |  |  |  |  |  |  |  |
| OUTPUT (cycles) |  |  |  |  |  |  |  |
| Basic slice header parsing | 9500 | 9500 | 9500 | 9500 | 9500 | 9500 | 9500 |
| Weighted prediction tables parsing | 1464 | 35136 | 1464 | 8784 | 8784 | 35136 | 35136 |
| Ref pic list modification parsing | 0 | 7460 | 0 | 7460 | 7460 | 7460 | 7460 |
| RPS parsing | 0 | 4420 | 0 | 4420 | 0 | 4420 | 0 |
| Ref pic re-ordering (operation) | 0 | 4800 | 0 | 4800 | 4800 | 4800 | 4800 |
| One slice header: total cycles | 10964 | 61316 | 10964 | 34964 | 30544 | 61316 | 56896 |
| **Total MHz for slice header parsing and operation** | **1** | **120** | **1** | **68** | **60** | **55** | **47** |

**Table 3. ARM9 cycle estimate for HEVC slice header parsing at 1080p@30 with proposed methods**

# Revised option 1 based on meeting discussion

It is proposed to impose the limit on the total number of luma\_weight\_l0\_flag, luma\_weight\_l1\_flag chroma\_weight\_l0\_flag and chroma\_weight\_l1\_flag that can be equal to 1 in pred\_weight\_table( ).

The suggested CD text change

|  |  |
| --- | --- |
| pred\_weight\_table( ) { | Descriptor |
| **luma\_log2\_weight\_denom** | ue(v) |
| if( chroma\_format\_idc != 0 ) |  |
| **delta\_chroma\_log2\_weight\_denom** | se(v) |
| if( slice\_type = = P | | slice\_type = = B ) |  |
| for( i = 0; i <= num\_ref\_idx\_l0\_active\_minus1; i++ ) { |  |
| **luma\_weight\_l0\_flag[i]** | u(1) |
| if( luma\_weight\_l0\_flag ) { |  |
| **delta\_luma\_weight\_l0[** i **]** | se(v) |
| **luma\_offset\_l0[** i **]** | se(v) |
| } |  |
| if( chroma\_format\_idc != 0 ) { |  |
| **chroma\_weight\_l0\_flag[i]** | u(1) |
| if( chroma\_weight\_l0\_flag ) |  |
| for( j =0; j < 2; j++ ) { |  |
| **delta\_chroma\_weight\_l0[** i **][** j **]** | se(v) |
| **delta\_chroma\_offset\_l0[** i **][** j **]** | se(v) |
| } |  |
| } |  |
| } |  |
| if( slice\_type = = B ) |  |
| for( i = 0; i <= num\_ref\_idx\_l1\_active\_minus1; i++ ) { |  |
| **luma\_weight\_l1\_flag[i]** | u(1) |
| if( luma\_weight\_l1\_flag ) { |  |
| **delta\_luma\_weight\_l1[** i **]** | se(v) |
| **luma\_offset\_l1[** i **]** | se(v) |
| } |  |
| if( chroma\_format\_idc != 0 ) { |  |
| **chroma\_weight\_l1\_flag[i]** | u(1) |
| if( chroma\_weight\_l1\_flag ) |  |
| for( j = 0; j < 2; j++ ) { |  |
| **delta\_chroma\_weight\_l1[** i **][** j **]** | se(v) |
| **delta\_chroma\_offset\_l1[** i **][** j **]** | se(v) |
| } |  |
| } |  |
| } |  |
| } |  |

#### Add the following constraint at the end of section 7.4.3.3 Weighted prediction parameters semantics(changes marked in yellow)

**luma\_log2\_weight\_denom** is the base 2 logarithm of the denominator for all luma weighting factors. The value of luma\_log2\_weight\_denom shall be in the range of 0 to 7, inclusive.

**delta\_chroma\_log2\_weight\_denom** is the difference of the base 2 logarithm of the denominator for all chroma weighting factors.

The variable ChromaLog2WeightDenom is specified by luma\_log2\_weight\_denom + delta\_chroma\_log2\_weight\_denom and it shall be in the range of 0 to 7, inclusive.

**luma\_weight\_l0\_flag[i]** equal to 1 specifies that weighting factors for the luma component of list 0 prediction are present. luma\_weight\_l0\_flag[i] equal to 0 specifies that these weighting factors are not present.

**delta\_luma\_weight\_l0[** i **]** is the difference of the weighting factor applied to the luma prediction value for list 0 prediction using RefPicList0[ i ].

The variable LumaWeightL0[ i ] is specified by (1 << luma\_log2\_weight\_denom ) + delta\_luma\_weight\_l0[ i ]. When luma\_weight\_l0\_flag is equal to 1, the value of LumaWeightL0[ i ] shall be in the range of −128 to 127, inclusive. When luma\_weight\_l0\_flagis equal to 0, LumaWeightL0[ i ] is inferred to be equal to 2luma\_log2\_weight\_denom for RefPicList0[ i ].

**luma\_offset\_l0[** i **]** is the additive offset applied to the luma prediction value for list 0 prediction using RefPicList0[ i ]. The value of luma\_offset\_l0[ i ] shall be in the range of −128 to 127, inclusive. When luma\_weight\_l0\_flagis equal to 0, luma\_offset\_l0[ i ] is inferred as equal to 0 for RefPicList0[ i ].

**chroma\_weight\_l0\_flag[i]** equal to 1 specifies that weighting factors for the chroma prediction values of list 0 prediction are present. chroma\_weight\_l0\_flag[i] equal to 0 specifies that these weighting factors are not present.

**delta\_chroma\_weight\_l0[** i **][** j **]** is the difference of the weighting factor applied to the chroma prediction values for list 0 prediction using RefPicList0[ i ] with j equal to 0 for Cb and j equal to 1 for Cr.

The variable ChromaWeightL0[ i ][ j ] is specified by ( 1 << ChromaLog2WeightDenom ) + delta\_chroma\_weight\_l0[ i ][ j ]. When chroma\_weight\_l0\_flag is equal to 1, the value of ChromaWeightL0[ i ][ j ] shall be in the range of −128 to 127, inclusive. When chroma\_weight\_l0\_flag is equal to 0**,** ChromaWeightL0[ i ][ j ] is inferred to be equal to 2ChromaLog2WeightDenom for RefPicList0[ i ].

**delta\_chroma\_offset\_l0[** i **][** j **]** is the difference of the additive offset applied to the chroma prediction values for list 0 prediction using RefPicList0[ i ] with j equal to 0 for Cb and j equal to 1 for Cr.

The variable ChromaOffsetL0[ i ][ j ] is specified as follows:

shift = 1 << ( BitDepthC − 1 )

ChromaOffsetL0[ i ][ j ] = (delta\_chroma\_offset\_l0[i][j] –   
 ( (shift\*ChromaWeightL0[ i ][ j ]) >> ChromaLog2WeightDenom ) − shift ) (7‑63)

The variable ChromaOffsetL0[ i ][ j ] shall be in the range of −127 to 128, inclusive. When chroma\_weight\_l0\_flag is equal to 0**,** ChromaOffsetL0[ i ][ j ] is inferred to be equal to 0 for RefPicList0[ i ].

**luma\_weight\_l1\_flag, delta\_luma\_weight\_l1**, **luma\_offset\_l1**, **chroma\_weight\_l1\_flag**, **delta\_chroma\_weight\_l1**, **delta\_chroma\_offset\_l1** have the same semantics as luma\_weight\_l0\_flag, delta\_luma\_weight\_l0, luma\_offset\_l0, chroma\_weight\_l0\_flag, delta\_chroma\_weight\_l0, delta\_chroma\_offset\_l0, respectively, with l0, list 0, and List0 replaced by l1, list 1, and List1, respectively.

For P slices, the sum value of luma\_weight\_l0\_flag[i], for i = 0, 1, …, num\_ref\_idx\_l0\_active\_minus1, shall be in the range from 0 to 8, inclusive.

For P slices, the sum value of chroma\_weight\_l0\_flag[i], for i = 0, 1, …, num\_ref\_idx\_l0\_active\_minus1, shall be in the range from 0 to 8, inclusive.

For B slices, the sum of sum\_luma\_weight\_l0\_flags and sum\_luma\_weight\_l1\_flags shall be in the range from 0 to 8, inclusive. Where sum\_luma\_weight\_l0\_flags is the sum value of luma\_weight\_l0\_flag[i], for i = 0, 1, …, num\_ref\_idx\_l0\_active\_minus1, and sum\_luma\_weight\_l1\_flags is the sum value of luma\_weight\_l1\_flag[i], for i = 0, 1, …, num\_ref\_idx\_l1\_active\_minus1.

For B slices, the sum of sum\_chroma\_weight\_l0\_flags and sum\_chroma\_weight\_l1\_flags shall be in the range from 0 to 8, inclusive. Where sum\_chroma\_weight\_l0\_flags is the sum value of chroma\_weight\_l0\_flag[i], for i = 0, 1, …, num\_ref\_idx\_l0\_active\_minus1, and sum\_chroma\_weight\_l1\_flags is the sum value of chroma\_weight\_l1\_flag[i], for i = 0, 1, …, num\_ref\_idx\_l1\_active\_minus1.

# Conclusions

The worst case HEVC slice parsing overhead is too high for a real-time decoder to handle. The proposed methods reduce worst case slice header parsing overhead by 2x to 3x when compared to the current design, thus significantly lowers decoder implementation burden. It is recommended to consider constraining worst slice header parsing overhead in the HEVC design to limit the ability of creating evil and facilitate the real-time HEVC applications.

# References

[1] F. Bossen, “Common test conditions and software reference configurations,” JCT-VC Document, JCTVC-I1100, 9th Meeting: Geneva, Switzerland, 27 April – 07 May, 2012

[2] [B. Bross](mailto:benjamin.bross@hhi.fraunhofer.de), [W.-J. Han](mailto:wjhan.han@samsung.com), [J.-R. Ohm](mailto:ohm@ient.rwth-aachen.de), [G. J. Sullivan](mailto:garysull@microsoft.com), [T. Wiegand](mailto:thomas.wiegand@hhi.fraunhofer.de) “High Efficiency Video Coding (HEVC) text specification draft 7,” JCT-VC Document, JCTVC-I1003, 9th Meeting: Geneva, Switzerland, 27 April – 07 May, 2012.

# Patent rights declaration(s)

**Texas Instruments, Inc. may have IPR relating to the technology described in this contribution and, conditioned on reciprocity, is prepared to grant licenses under reasonable and non-discriminatory terms as necessary for implementation of the resulting ITU-T Recommendation |ISO/IEC International Standard (per box 2 of the ITU-T/ITU-R/ISO/IEC patent statement and licensing declaration form).**

# CD text

CD text for moving RPS to APS and option 1 of cycle overhead reduction for weighted prediction table.

Changes marked as yellow

#### Replace 7.3.2.6 Adaptation parameter set RBSP syntax

with

|  |  |  |
| --- | --- | --- |
| aps\_rbsp( ) { | Descriptor | |
| **aps\_id** | ue(v) | | | |
| **aps\_ref\_pic\_set\_present\_flag** | | u(1) |
| if (aps\_ref\_pic\_set\_present\_flag ) { | |  |
| **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 ) { | |  | |
| **num\_long\_term\_pics** | | ue(v) | |
| for( i = 0; i < num\_long\_term\_pics; i++ ) { | |  | |
| **poc\_lsb\_lt**[ i ] | | u(v) | |
| **delta\_poc\_msb\_present\_flag**[ i ] | | u(1) | |
| if( delta\_poc\_msb\_present\_flag[ i ] ) | |  | |
| **delta\_poc\_msb\_cycle\_lt**[ i ] | | ue(v) | |
| **used\_by\_curr\_pic\_lt\_flag**[ i ] | | u(1) | |
| } | |  | |
| } | |  | |
| } | |  | |
| **aps\_adaptive\_loop\_filter\_flag** | u(1) | |
| if( aps\_adaptive\_loop\_filter\_flag ) |  | |
| alf\_param( ) |  | |
| **aps\_extension\_flag** | u(1) | |
| if( aps\_extension\_flag ) |  | |
| while( more\_rbsp\_data( ) ) |  | |
| **aps\_extension\_data\_flag** | u(1) | |
| rbsp\_trailing\_bits( ) |  | |
| } |  | |

Replace 7.4.2.6 **Adaptation parameter set RBSP semantics with**

**aps\_id** identifies the adaptation parameter set that is referred to in the slice header. The value of aps\_id shall be in the range of 0 to TBD, inclusive.

[Ed. (BB): Proponent suggests that this could be made level/profile dependend. ]

**aps\_ref\_pic\_set\_present\_flag** equal to 1 specifies that the reference picture set is present in the APS; equal to 0 specifies that the reference picture set is not present in the APS. aps\_ref\_pic\_set\_present\_flag shall be 1 if IdrPicFlag is set to 0.

**pic\_order\_cnt\_lsb** specifies the picture order count modulo MaxPicOrderCntLsb for the current picture. The length of the pic\_order\_cnt\_lsb syntax element is log2\_max\_pic\_order\_cnt\_lsb\_minus4 + 4 bits. The value of the pic\_order\_cnt\_lsb shall be in the range of 0 to MaxPicOrderCntLsb − 1, inclusive. When pic\_order\_cnt\_lsb is not present, pic\_order\_cnt\_lsb is inferred to be equal to 0.

**short\_term\_ref\_pic\_set\_sps\_flag** equal to 1 specifies that the short-term reference picture set of the current picture shall be created using syntax elements in the active sequence parameter set. short\_term\_ref\_pic\_set\_sps\_flag equal to 0 specifies that the short-term reference picture set of the current picture shall be created using syntax elements in the short\_term\_ref\_pic\_set( ) syntax structure in the slice header.

**short\_term\_ref\_pic\_set\_idx** specifies the index to the list of the short-term reference picture sets specified in the active sequence parameter set that shall be used for creation of the reference picture set of the current picture. The syntax element short\_term\_ref\_pic\_set\_idx shall be represented by Ceil( Log2( num\_short\_term\_ref\_pic\_sets ) ) bits. The value of short\_term\_ref\_pic\_set\_idx shall be in the range of 0 to num\_short\_term\_ref\_pic\_sets − 1, inclusive, where num\_short\_term\_ref\_pic\_sets is the syntax element from the active sequence parameter set.

The variable StRpsIdx is derived as follows.

if( short\_term\_ref\_pic\_set\_sps\_flag )  
 StRpsIdx = short\_term\_ref\_pic\_set\_idx (7‑36)  
else  
 StRpsIdx = num\_short\_term\_ref\_pic\_sets

**num\_long\_term\_pics** specifies the number of the long-term reference pictures that are to be included in the long-term reference picture set of the current picture. The value of num\_long\_term\_pics shall be in the range of 0 to sps\_max\_dec\_pic\_buffering[ sps\_max\_temporal\_layers\_minus1 ] – NumNegativePics[ StRpsIdx ] – NumPositivePics[ StRpsIdx ] , inclusive. When not present, the value of num\_long\_term\_pics is inferred to be equal to 0.

**poc\_lsb\_lt**[ i ] specifies the value of the least significant bits of the picture order count value of the i-th long-term reference picture that is included in the long-term reference picture set of the current picture. The length of the poc\_lsb\_lt[ i ] syntax element is log2\_max\_pic\_order\_cnt\_lsb\_minus4 + 4 bits. For any values of j and k in the range of 0 to num\_long\_term\_pics – 1, inclusive, if j is less than k, poc\_lsb\_lt[ j ] shall not be less than poc\_lsb\_lt[ k ].

**delta\_poc\_msb\_present\_flag[ i ]** equal to 1 specifies that delta\_poc\_msb\_cycle\_lt\_minus1[ i ] is present. delta\_poc\_msb\_present\_flag[ i ]equal to 0 specifies that delta\_poc\_msb\_cycle\_lt\_minus1[ i ] is not present. delta\_poc\_msb\_present\_flag[ i ] shall be equal to 1 when there is more than one reference picture in the decoded picture buffer with the least significant bits of the picture order count value equal to poc\_lsb\_lt[ i ].

**delta\_poc\_msb\_cycle\_lt**[ i ]is used to determine the value of the most significant bits of the picture order count value of the i-th long-term reference picture that is included in the long-term reference picture set of the current picture.

The variable DeltaPocMSBCycleLt[ i ] is derived as follows.

if( i = = 0 | | poc\_lsb\_lt[ i − 1 ] ! = poc\_lsb\_lt[ i ] )   
 DeltaPocMSBCycleLt[ i ] = delta\_poc\_msb\_cycle\_lt[ i ]   
 else (7‑37)  
 DeltaPocMSBCycleLt[ i ] = delta\_poc\_msb\_cycle\_lt[ i ] + DeltaPocMSBCycleLt[ i − 1 ]

The value of DeltaPocMSBCycleLt[ i ] \* MaxPicOrderCntLsb + pic\_order\_cnt\_lsb – poc\_lsb\_lt[ i ] shall be in the range of 1 to 224 – 1, inclusive.

**used\_by\_curr\_pic\_lt\_flag**[ i ] equal to 0 specifies that the i-th long-term reference picture included in the long-term reference picture set of the current picture is not used for reference by the current picture.

**aps\_adaptive\_loop\_filter\_flag** equal to 1 specifies that the ALF is on for slices referring to the current APS; equal to 0 specifies that the ALF is off for slices referring to the current APS. When there is no active APS, the aps\_adaptive\_loop\_filter\_flag value is inferred to be 0.

**aps\_extension\_flag** equal to 0 specifies that no aps\_extension\_data\_flag syntax elements are present in the picture parameter set RBSP syntax structure. aps\_extension\_flag shall be equal to 0 in bitstreams conforming to this Recommendation | International Standard. The value of 1 for aps\_extension\_flag is reserved for future use by ITU‑T | ISO/IEC. Decoders shall ignore all data that follow the value 1 for aps\_extension\_flag in a picture parameter set NAL unit.

**aps\_extension\_data\_flag** may have any value. Its value does not affect decoder conformance to profiles specified in this Recommendation | International Standard.

Replace 7.3.3 Slice header syntax

With

|  |  |
| --- | --- |
| slice\_header( ) { | Descriptor |
| **first\_slice\_in\_pic\_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( RapPicFlag ) { |  |
| **rap\_pic\_id** | ue(v) |
| **no\_output\_of\_prior\_pics\_flag** | u(1) |
| } |  |
| ~~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 ) {~~ |  |
| **~~num\_long\_term\_pics~~** | ~~ue(v)~~ |
| ~~for( i = 0; i < num\_long\_term\_pics; i++ ) {~~ |  |
| **~~poc\_lsb\_lt~~**~~[ i ]~~ | ~~u(v)~~ |
| **~~delta\_poc\_msb\_present\_flag~~**~~[ i ]~~ | ~~u(1)~~ |
| ~~if( delta\_poc\_msb\_present\_flag[ i ] )~~ |  |
| **~~delta\_poc\_msb\_cycle\_lt~~**~~[ i ]~~ | ~~ue(v)~~ |
| **~~used\_by\_curr\_pic\_lt\_flag~~**~~[ i ]~~ | ~~u(1)~~ |
| ~~}~~ |  |
| ~~}~~ |  |
| ~~}~~ |  |
| if( sample\_adaptive\_offset\_enabled\_flag ) { |  |
| **slice\_sample\_adaptive\_offset\_flag[** 0 **]** | u(1) |
| if( slice\_sample\_adaptive\_offset\_flag[ 0 ] ) { |  |
| **slice\_sample\_adaptive\_offset\_flag[** 1 **]** | u(1) |
| **slice\_sample\_adaptive\_offset\_flag[** 2 **]** | u(1) |
| } |  |
| } |  |
| if(adaptive\_loop\_filter\_enabled\_flag ||  ! IdrPicFlag ) |  |
| **aps\_id** | ue(v) |
| if( slice\_type = = P | | slice\_type = = B ) { |  |
| if( sps\_temporal\_mvp\_enable\_flag ) |  |
| **pic\_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 && slice\_type != I ) |  |
| **cabac\_init\_flag** | u(1) |
| **slice\_qp\_delta** | 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( pic\_temporal\_mvp\_enable\_flag ) { |  |
| if( slice\_type = = B ) |  |
| **collocated\_from\_l0\_flag** | u(1) |
| if( slice\_type != I &&   ((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) |
| } |  |
| if( ( weighted\_pred\_flag && slice\_type = = P) | |  ( weighted\_bipred\_idc = = 1 && slice\_type = = B ) ) |  |
| pred\_weight\_table( ) |  |
| if( slice\_type = = P | | slice\_type = = B ) |  |
| **five\_minus\_max\_num\_merge\_cand** | ue(v) |
| if( adaptive\_loop\_filter\_enabled\_flag ) { |  |
| **slice\_adaptive\_loop\_filter\_flag** | u(1) |
| if( slice\_adaptive\_loop\_filter\_flag && alf\_coef\_in\_slice\_flag ) |  |
| alf\_param( ) |  |
| if( slice\_adaptive\_loop\_filter\_flag && !alf\_coef\_in\_slice\_flag ) |  |
| alf\_cu\_control\_param( ) |  |
| } |  |
| if( seq\_loop\_filter\_across\_slices\_enabled\_flag &&  ( slice\_adaptive\_loop\_filter\_flag | | slice\_sample\_adaptive\_offset\_flag | |  !disable\_deblocking\_filter\_flag ) ) |  |
| **slice\_loop\_filter\_across\_slices\_enabled\_flag** | u(1) |
| } |  |
| if( tiles\_or\_entropy\_coding\_sync\_idc = = 1 | |  tiles\_or\_entropy\_coding\_sync\_idc = = 2 ) { |  |
| **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** | u(8) |
| } |  |
| byte\_alignment( ) |  |
| } |  |

### Replace 7.4.3 Slice header semantics with

When present, the value of the slice header syntax elements pic\_parameter\_set\_id, pic\_output\_flag, rap\_pic\_id, no\_output\_of\_prior\_pics\_flag, ~~pic\_order\_cnt\_lsb, short\_term\_ref\_pic\_set\_sps\_flag, short\_term\_ref\_pic\_set\_idx, num\_long\_term\_pics and pic\_temporal\_mvp\_enable\_flag~~ shall be the same in all slice headers of a coded picture. ~~When present, the value of the slice header syntax elements poc\_lsb\_lt[ i ], delta\_poc\_msb\_present\_flag[ i ], delta\_poc\_msb\_cycle\_lt[ i ] and used\_by\_curr\_pic\_lt\_flag[ i ] shall be the same in all slice headers of a coded picture for each i in the range of 0 to num\_long\_term\_pics, inclusive.~~

**first\_slice\_in\_pic\_flag** indicates whether the slice is the first slice of the picture. If first\_slice\_in\_pic\_flag is equal to 1, the variables SliceCbAddrZS and SliceCtbAddrRS are both set to 0 and the decoding starts with the first coding tree block in the picture.

**pic\_parameter\_set\_id** specifies the picture parameter set in use. The value of pic\_parameter\_set\_id shall be in the range of 0 to 255, inclusive.

**slice\_address** specifies the address in slice granularity resolution in which the slice starts. The length of the slice\_address syntax element is ( Ceil( Log2( PicWidthInCtbs \* PicHeightInCtbs ) ) + SliceGranularity ) bits.

The variable SliceCtbAddrRS, specifying the coding tree block in which the slice starts in coding tree block raster scan order, is derived as follows.

SliceCtbAddrRS = ( slice\_address >> SliceGranularity ) (7‑34)

The variable SliceCbAddrZS, specifying the address of first coding block in the slice in minimum coding block granularity in z-scan order, is derived as follows.

SliceCbAddrZS = slice\_address << ( ( log2\_diff\_max\_min\_coding\_block\_size − SliceGranularity ) <<1 ) (7‑35)

[Ed. (BB): Change it that SliceCbAddrZS is in z-scan order in a coding tree block while the CTBs are in tiles scan order]

The slice decoding starts with the largest coding unit possible at the slice starting coordinate. [Ed. (BB): More precise description is needed here.]

**dependent\_slice\_flag** equal to 1 specifies that the value of each slice header syntax element not present is inferred to be equal to the value of corresponding slice header syntax element in the preceding slice containing the coding tree block for which the coding tree block address is SliceCtbAddrRS − 1. When not present, the value of dependent\_slice\_flag is inferred to be equal to 0. The value of dependent\_slice\_flag shall be equal to 0 when SliceCtbAddrRS equal to 0.

**slice\_type** specifies the coding type of the slice according to .

Table 7‑7 – Name association to slice\_type

|  |  |
| --- | --- |
| slice\_type | Name of slice\_type |
| 0 | B (B slice) |
| 1 | P (P slice) |
| 2 | I (I slice) |

When nal\_unit\_type is equal to a value in the range of 4 to 8, inclusive (RAP picture), slice\_type shall be equal to 2.

When sps\_max\_dec\_pic\_buffering[ sps\_max\_temporal\_layers\_minus1 ] is equal to 0, slice\_type shall be equal to 2.

**colour\_plane\_id** specifies the colour plane associated with the current slice RBSP when separate\_colour\_plane\_flag is equal to 1. The value of colour\_plane\_id shall be in the range of 0 to 2, inclusive. colour\_plane\_id equal to 0, 1, and 2 correspond to the Y, Cb, and Cr planes, respectively.

NOTE 1 – There is no dependency between the decoding processes of pictures having different values of colour\_plane\_id.

**pic\_output\_flag** affects the decoded picture output and removal processes as specified in Annex C. When pic\_output\_flag is not present, it is inferred to be equal to 1.

**rap\_pic\_id** identifies a RAP picture. The values of rap\_pic\_id in all the slices of a RAP picture shall remain unchanged. When two consecutive access units in decoding order are both RAP access units, the value of rap\_pic\_id in the slices of the first such RAP access unit shall differ from the rap\_pic\_id in the second such RAP access unit. The value of rap\_pic\_id shall be in the range of 0 to 65535, inclusive.

**no\_output\_of\_prior\_pics\_flag** specifies how the previously-decoded pictures in the decoded picture buffer are treated after decoding of an IDR or a BLA picture. See Annex . When the IDR or BLA picture is the first IDR or BLA picture in the bitstream, the value of no\_output\_of\_prior\_pics\_flag has no effect on the decoding process. When the IDR or BLA picture is not the first IDR or BLA picture in the bitstream and the value of pic\_width\_in\_luma\_samples or pic\_height\_in\_luma\_samples or sps\_max\_dec\_pic\_buffering[ sps\_max\_temporal\_layers\_minus1 ] derived from the active sequence parameter set is different from the value of pic\_width\_in\_luma\_samples or pic\_height\_in\_luma\_samples or sps\_max\_dec\_pic\_buffering[ sps\_max\_temporal\_layers\_minus1 ] derived from the sequence parameter set active for the preceding picture, no\_output\_of\_prior\_pics\_flag equal to 1 may (but should not) be inferred by the decoder, regardless of the actual value of no\_output\_of\_prior\_pics\_flag.

**~~pic\_order\_cnt\_lsb~~** ~~specifies the picture order count modulo MaxPicOrderCntLsb for the current picture. The length of the pic\_order\_cnt\_lsb syntax element is log2\_max\_pic\_order\_cnt\_lsb\_minus4 + 4 bits. The value of the pic\_order\_cnt\_lsb shall be in the range of 0 to MaxPicOrderCntLsb − 1, inclusive. When pic\_order\_cnt\_lsb is not present, pic\_order\_cnt\_lsb is inferred to be equal to 0.~~

**~~short\_term\_ref\_pic\_set\_sps\_flag~~** ~~equal to 1 specifies that the short-term reference picture set of the current picture shall be created using syntax elements in the active sequence parameter set. short\_term\_ref\_pic\_set\_sps\_flag equal to 0 specifies that the short-term reference picture set of the current picture shall be created using syntax elements in the short\_term\_ref\_pic\_set( ) syntax structure in the slice header.~~

**~~short\_term\_ref\_pic\_set\_idx~~** ~~specifies the index to the list of the short-term reference picture sets specified in the active sequence parameter set that shall be used for creation of the reference picture set of the current picture. The syntax element short\_term\_ref\_pic\_set\_idx shall be represented by Ceil( Log2( num\_short\_term\_ref\_pic\_sets ) ) bits. The value of short\_term\_ref\_pic\_set\_idx shall be in the range of 0 to num\_short\_term\_ref\_pic\_sets − 1, inclusive, where num\_short\_term\_ref\_pic\_sets is the syntax element from the active sequence parameter set.~~

~~The variable StRpsIdx is derived as follows.~~

~~if( short\_term\_ref\_pic\_set\_sps\_flag )  
 StRpsIdx = short\_term\_ref\_pic\_set\_idx (7‑36)  
else  
 StRpsIdx = num\_short\_term\_ref\_pic\_sets~~

**~~num\_long\_term\_pics~~** ~~specifies the number of the long-term reference pictures that are to be included in the long-term reference picture set of the current picture. The value of num\_long\_term\_pics shall be in the range of 0 to sps\_max\_dec\_pic\_buffering[ sps\_max\_temporal\_layers\_minus1 ] – NumNegativePics[ StRpsIdx ] – NumPositivePics[ StRpsIdx ] , inclusive. When not present, the value of num\_long\_term\_pics is inferred to be equal to 0.~~

**~~poc\_lsb\_lt~~**~~[ i ] specifies the value of the least significant bits of the picture order count value of the i-th long-term reference picture that is included in the long-term reference picture set of the current picture. The length of the poc\_lsb\_lt[ i ] syntax element is log2\_max\_pic\_order\_cnt\_lsb\_minus4 + 4 bits. For any values of j and k in the range of 0 to num\_long\_term\_pics – 1, inclusive, if j is less than k, poc\_lsb\_lt[ j ] shall not be less than poc\_lsb\_lt[ k ].~~

**~~delta\_poc\_msb\_present\_flag[ i ]~~** ~~equal to 1 specifies that delta\_poc\_msb\_cycle\_lt\_minus1[ i ] is present. delta\_poc\_msb\_present\_flag[ i ]~~~~equal to 0 specifies that delta\_poc\_msb\_cycle\_lt\_minus1[ i ] is not present. delta\_poc\_msb\_present\_flag[ i ] shall be equal to 1 when there is more than one reference picture in the decoded picture buffer with the least significant bits of the picture order count value equal to poc\_lsb\_lt[ i ].~~

**~~delta\_poc\_msb\_cycle\_lt~~**~~[ i ]~~~~is used to determine the value of the most significant bits of the picture order count value of the i-th long-term reference picture that is included in the long-term reference picture set of the current picture.~~

~~The variable DeltaPocMSBCycleLt[ i ] is derived as follows.~~

~~if( i = = 0 | | poc\_lsb\_lt[ i − 1 ] ! = poc\_lsb\_lt[ i ] )   
 DeltaPocMSBCycleLt[ i ] = delta\_poc\_msb\_cycle\_lt[ i ]   
 else (7‑37)  
 DeltaPocMSBCycleLt[ i ] = delta\_poc\_msb\_cycle\_lt[ i ] + DeltaPocMSBCycleLt[ i − 1 ]~~

~~The value of DeltaPocMSBCycleLt[ i ] \* MaxPicOrderCntLsb + pic\_order\_cnt\_lsb – poc\_lsb\_lt[ i ] shall be in the range of 1 to 2~~~~24~~~~– 1, inclusive.~~

**~~used\_by\_curr\_pic\_lt\_flag~~**~~[ i ] equal to 0 specifies that the i-th long-term reference picture included in the long-term reference picture set of the current picture is not used for reference by the current picture.~~

**pic\_temporal\_mvp\_enable\_flag** specifies whether temporal motion vector predictors can be used for inter prediction. If pic\_temporal\_mvp\_enable\_flag is equal to 0, the temporal motion vector predictors shall not be used in decoding of the current picture. If pic\_temporal\_mvp\_enable\_flag is equal to 1, temporal motion vector predictors may be used in decoding of the current picture. When not present, the value of pic\_temporal\_mvp\_enable\_flag shall be inferred to be equal to 0.

When both pic\_temporal\_mvp\_enable\_flag and temporal\_id are equal to 0, all coded pictures that follow the current picture in decoding order shall not use temporal motion vector vectors from any picture that precedes the current picture in decoding order.

**num\_ref\_idx\_active\_override\_flag** equal to 1 specifies that the syntax element num\_ref\_idx\_l0\_active\_minus1 is present for P and B slices and that the syntax element num\_ref\_idx\_l1\_active\_minus1 is present for B slices. num\_ref\_idx\_active\_override\_flag equal to 0 specifies that the syntax elements num\_ref\_idx\_l0\_active\_minus1 and num\_ref\_idx\_l1\_active\_minus1 are not present.

**num\_ref\_idx\_l0\_active\_minus1** specifies the maximum reference index for reference picture list 0 that may be used to decode the slice. When the current slice is a P slice, num\_ref\_idx\_l0\_active\_minus1 shall be in the range of 0 to 7, inclusive. When the current slice is a B slice, num\_ref\_idx\_l0\_active\_minus1 + num\_ref\_idx\_l1\_active\_minus1 shall be in the range of 0 to 6, inclusive. [Ed. (GJS): Constrain to the value imposed by the max DPB size limit instead of 15. (YK): It should be allowed to use the max number of active entries greater than the max DPB size limit as part of the level definition.]

When the current slice is a P or B slice and num\_ref\_idx\_l0\_active\_minus1 is not present, num\_ref\_idx\_l0\_active\_minus1 is inferred to be equal to num\_ref\_idx\_l0\_default\_active\_minus1.

**num\_ref\_idx\_l1\_active\_minus1** specifies the maximum reference index for reference picture list 1 that shall be used to decode the slice. num\_ref\_idx\_l0\_active\_minus1 + num\_ref\_idx\_l1\_active\_minus1 shall be in the range of 0 to 6, inclusive.

When the current slice is a B slice and num\_ref\_idx\_l1\_active\_minus1 is not present, num\_ref\_idx\_l1\_active\_minus1 is inferred to be equal to num\_ref\_idx\_l1\_default\_active\_minus1.

~~The range of num\_ref\_idx\_l1\_active\_minus1 is constrained as specified in the semantics for num\_ref\_idx\_l0\_active\_minus1 with l0 and list 0 replaced by l1 and list 1, respectively.~~

**mvd\_l1\_zero\_flag** equal to 1 indicates that difference between a list 1 vector component and its prediction, mvd\_l1[][][], is not parsed and it is set equal to 0. mvd\_l1\_zero\_flag equal to 0 indicates that mvd\_l1[][][] is parsed.

**cabac\_init\_flag** specifies the method for determining the initialisation table used in the initialisation process for context variables. The value of cabac\_init\_flag shall be in the range of 0 to 1, inclusive. When cabac\_init\_flag is not present, it is inferred to be 0.

**slice\_qp\_delta** specifies the initial value of QPY to be used for all the coding blocks in the slice until modified by the value of cu\_qp\_delta in the coding unit layer. The initial QPY quantization parameter for the slice is computed as

SliceQPY = 26 + pic\_init\_qp\_minus26 + slice\_qp\_delta (7‑38)

The value of slice\_qp\_delta shall be limited such that SliceQPY is in the range of −QpBdOffsetY to +51, inclusive.

**deblocking\_filter\_override\_flag** equal to 0 specifies that deblocking parameters from the active picture parameter set shall be used for deblocking the current slice. deblocking\_filter\_override\_flag equal to 0 specifies that deblocking parameters from the slice header shall be used for deblocking the current slice. When not present, the value of deblocking\_filter\_override\_flag is inferred to be equal to 1.

**slice\_header\_disable\_deblocking\_filter\_flag** equal to 1 specifies that the operation of the deblocking filter shall not be applied for the current slice. slice\_header\_disable\_deblocking\_filter\_flag equal to 0 specifies that the operation of the deblocking filter shall be applied for the current slice.

**beta\_offset\_div2** and **tc\_offset\_div2** specify the deblocking parameter offsets for β and tC (divided by 2) for the current slice.

**slice\_sample\_adaptive\_offset\_flag[** cIdx **]** equal to 1 specifies that SAO is enabled for the colour component cIdx in the current slice; equal to 0 specifies that SAO is disabled for the colour component cIdx in the current slice. When slice\_sample\_adaptive\_offset\_flag[ cIdx ] is not present, slice\_sample\_adaptive\_offset\_flag[ cIdx ] is inferred to be equal to 0.

**collocated\_from\_l0\_flag** equal to 1 specifies the picture that contains the collocated partition shall be derived from list 0, otherwise the picture shall be derived from list 1.When collocated\_from\_l0\_flag is not present, it is inferred to be equal to 1.

**collocated\_ref\_idx** specifies the reference index of the picture that contains the collocated partition. When the current slice is a P slice collocated\_ref\_idx refers to list 0. When the current slice is a B slice collocated\_ref\_idx refers to list 0 if collocated\_from\_l0 is 1, otherwise it refers to list 1. Collocated\_ref\_idx shall always refer to a valid list entry, and the resulting picture shall be the same for all slices of a coded picture. When collocated\_ref\_idx is not present, it is inferred to be equal to 0.

**five\_minus\_max\_num\_merge\_cand** specifies the maximum number of merging MVP candidates supported in the slice subtracted from 5. The maximum number of merging MVP candidates, MaxNumMergeCand is computed as

MaxNumMergeCand = 5 − five\_minus\_max\_num\_merge\_cand (7‑39)

The value of five\_minus\_max\_num\_merge\_cand shall be limited such that MaxNumMergeCand is in the range of 0 to 5, inclusive.

**slice\_adaptive\_loop\_filter\_flag** equal to 1 specifies that the ALF is on for the current slice; equal to 0 specifies that the ALF is off for the current slice. It is a requirement of the bitstream that if there is an active APS, the value of slice\_adaptive\_loop\_filter\_flag shall be equal to aps\_adaptive\_loop\_filter\_flag.

**slice\_loop\_filter\_across\_slices\_enabled\_flag** equal to 1 specifies that in-loop filtering operations are performed across slice boundaries; otherwise, the in-loop operations are slice-independent and not applied across slice boundaries. The in-loop filtering operations include the deblocking filter, sample adaptive offset filter, and adaptive loop filter. When slice\_loop\_filter\_across\_slices\_enabled\_flag is not present, it is inferred to be equal to seq\_loop\_filter\_across\_slices\_enabled\_flag.

**num\_entry\_point\_offsets** specifies the number of entry\_point\_offset[ i ] syntax elements in the slice header. When tiles\_or\_entropy\_coding\_sync\_idc is equal to 1, the value of num\_entry\_point\_offsets shall be in the range of 0 to ( num\_tile\_columns\_minus1 + 1 ) \* ( num\_tile\_rows\_minus1 + 1 ) − 1, inclusive. When tiles\_or\_entropy\_coding\_sync\_idc is equal to 2, the value of num\_entry\_point\_offsets shall be in the range of 0 to PicHeightInCtbs − 1, inclusive. When not present, the value of num\_entry\_point\_offsets is inferred to be equal to 0.

**offset\_len\_minus1** plus 1 specifies the length, in bits, of the entry\_point\_offset[ i ] syntax elements.

**entry\_point\_offset**[ i ] specifies the i-th entry point offset, in bytes and shall be represented by offset\_len\_minus1 plus 1 bits. The coded slice data after the slice header consists of num\_entry\_point\_offsets + 1 subsets, with subset index values ranging from 0 to num\_entry\_point\_offsets, inclusive. Subset 0 consists of bytes 0 to entry\_point\_offset[ 0 ] − 1, inclusive, of the coded slice data, subset k, with k in the range of 1 to num\_entry\_point\_offsets - 1, inclusive, consists of bytes entry\_point\_offset[ k − 1 ] to entry\_point\_offset[ k ] + entry\_point\_offset[ k − 1 ] − 1, inclusive, of the coded slice data, and the last subset (with subset index equal to num\_entry\_point\_offsets) consists of the remaining bytes of the coded slice data.

When tiles\_or\_entropy\_coding\_sync\_idc is equal to 1 and num\_entry\_point\_offsets is greater than 0, each subset shall contain all coded bits of exactly one tile, and the number of subsets (i.e., the value of num\_entry\_point\_offsets + 1) shall be equal to or less than the number of tiles in the slice.

NOTE 2 – When tiles\_or\_entropy\_coding\_sync\_idc is equal to 1, each slice must include either a subset of one tile (in which case signalling of entry points is unnecessary) or an integer number of complete tiles.

When tiles\_or\_entropy\_coding\_sync\_idc is equal to 2 and num\_entry\_point\_offsets is greater than 0, each subset k with k in the range of 0 to num\_entry\_point\_offsets − 1, inclusive, shall contain all coded bits of exactly one row of coding tree blocks, the last subset (with subset index equal to num\_entry\_point\_offsets) shall contain all coded bits of the remaining coding blocks included in the slice, wherein the remaining coding blocks consist of either exactly one row of coding tree blocks or a subset of one row of coding tree blocks, and the number of subsets (i.e., the value of num\_entry\_point\_offsets + 1) shall be equal to the number of rows of coding tree blocks in the slice, wherein a subset of one row of coding tree blocks in the slice is also counted..

NOTE 3 – When tiles\_or\_entropy\_coding\_sync\_idc is equal to 2, a slice may include a number of rows of coding tree blocks and a subset of a row of coding tree blocks. For example, if a slice include two and a half rows of coding tree blocks, the the number of subsets (i.e., the value of num\_entry\_point\_offsets + 1) shall be equal to 3.

**slice\_header\_extension\_length** specifies the length of the slice header extension data in bytes, not including the bits used for signalling slice\_header\_extension\_length itself. The value of slice\_header\_extension\_length shall be in the range of 0 to 256, inclusive.

**slice\_header\_extension\_data\_byte** may have any value. Decoders shall ignore the value of slice\_header\_extension\_data\_byte. Its value does not affect decoder conformance to profiles specified in this Recommendation | International Standard.

CD text for moving RPS to APS and option 2 of cycle overhead reduction for weighted prediction table.

Changes marked as yellow

#### Replace 7.3.2.6 Adaptation parameter set RBSP syntax

with

|  |  |  |
| --- | --- | --- |
| aps\_rbsp( ) { | Descriptor | |
| **aps\_id** | ue(v) | | | |
| **aps\_ref\_pic\_set\_present\_flag** | | u(1) |
| if (aps\_ref\_pic\_set\_present\_flag ) { | |  |
| **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 ) { | |  | |
| **num\_long\_term\_pics** | | ue(v) | |
| for( i = 0; i < num\_long\_term\_pics; i++ ) { | |  | |
| **poc\_lsb\_lt**[ i ] | | u(v) | |
| **delta\_poc\_msb\_present\_flag**[ i ] | | u(1) | |
| if( delta\_poc\_msb\_present\_flag[ i ] ) | |  | |
| **delta\_poc\_msb\_cycle\_lt**[ i ] | | ue(v) | |
| **used\_by\_curr\_pic\_lt\_flag**[ i ] | | u(1) | |
| } | |  | |
| } | |  | |
| } | |  | |
| **aps\_pred\_weight\_table\_present\_flag** | | u(1) | | |
| if (aps\_pred\_weight\_table\_present\_flag) | |  | | |
| pred\_weight\_table( ) | |  | | |
| **aps\_adaptive\_loop\_filter\_flag** | u(1) | |
| if( aps\_adaptive\_loop\_filter\_flag ) |  | |
| alf\_param( ) |  | |
| **aps\_extension\_flag** | u(1) | |
| if( aps\_extension\_flag ) |  | |
| while( more\_rbsp\_data( ) ) |  | |
| **aps\_extension\_data\_flag** | u(1) | |
| rbsp\_trailing\_bits( ) |  | |
| } |  | |

Replace 7.4.2.6 **Adaptation parameter set RBSP semantics with**

**aps\_id** identifies the adaptation parameter set that is referred to in the slice header. The value of aps\_id shall be in the range of 0 to TBD, inclusive.

[Ed. (BB): Proponent suggests that this could be made level/profile dependend. ]

**aps\_ref\_pic\_set\_present\_flag** equal to 1 specifies that the reference picture set is present in the APS; equal to 0 specifies that the reference picture set is not present in the APS. aps\_ref\_pic\_set\_present\_flag shall be 1 if IdrPicFlag is set to 0.

**pic\_order\_cnt\_lsb** specifies the picture order count modulo MaxPicOrderCntLsb for the current picture. The length of the pic\_order\_cnt\_lsb syntax element is log2\_max\_pic\_order\_cnt\_lsb\_minus4 + 4 bits. The value of the pic\_order\_cnt\_lsb shall be in the range of 0 to MaxPicOrderCntLsb − 1, inclusive. When pic\_order\_cnt\_lsb is not present, pic\_order\_cnt\_lsb is inferred to be equal to 0.

**short\_term\_ref\_pic\_set\_sps\_flag** equal to 1 specifies that the short-term reference picture set of the current picture shall be created using syntax elements in the active sequence parameter set. short\_term\_ref\_pic\_set\_sps\_flag equal to 0 specifies that the short-term reference picture set of the current picture shall be created using syntax elements in the short\_term\_ref\_pic\_set( ) syntax structure in the slice header.

**short\_term\_ref\_pic\_set\_idx** specifies the index to the list of the short-term reference picture sets specified in the active sequence parameter set that shall be used for creation of the reference picture set of the current picture. The syntax element short\_term\_ref\_pic\_set\_idx shall be represented by Ceil( Log2( num\_short\_term\_ref\_pic\_sets ) ) bits. The value of short\_term\_ref\_pic\_set\_idx shall be in the range of 0 to num\_short\_term\_ref\_pic\_sets − 1, inclusive, where num\_short\_term\_ref\_pic\_sets is the syntax element from the active sequence parameter set.

The variable StRpsIdx is derived as follows.

if( short\_term\_ref\_pic\_set\_sps\_flag )  
 StRpsIdx = short\_term\_ref\_pic\_set\_idx (7‑36)  
else  
 StRpsIdx = num\_short\_term\_ref\_pic\_sets

**num\_long\_term\_pics** specifies the number of the long-term reference pictures that are to be included in the long-term reference picture set of the current picture. The value of num\_long\_term\_pics shall be in the range of 0 to sps\_max\_dec\_pic\_buffering[ sps\_max\_temporal\_layers\_minus1 ] – NumNegativePics[ StRpsIdx ] – NumPositivePics[ StRpsIdx ] , inclusive. When not present, the value of num\_long\_term\_pics is inferred to be equal to 0.

**poc\_lsb\_lt**[ i ] specifies the value of the least significant bits of the picture order count value of the i-th long-term reference picture that is included in the long-term reference picture set of the current picture. The length of the poc\_lsb\_lt[ i ] syntax element is log2\_max\_pic\_order\_cnt\_lsb\_minus4 + 4 bits. For any values of j and k in the range of 0 to num\_long\_term\_pics – 1, inclusive, if j is less than k, poc\_lsb\_lt[ j ] shall not be less than poc\_lsb\_lt[ k ].

**delta\_poc\_msb\_present\_flag[ i ]** equal to 1 specifies that delta\_poc\_msb\_cycle\_lt\_minus1[ i ] is present. delta\_poc\_msb\_present\_flag[ i ]equal to 0 specifies that delta\_poc\_msb\_cycle\_lt\_minus1[ i ] is not present. delta\_poc\_msb\_present\_flag[ i ] shall be equal to 1 when there is more than one reference picture in the decoded picture buffer with the least significant bits of the picture order count value equal to poc\_lsb\_lt[ i ].

**delta\_poc\_msb\_cycle\_lt**[ i ]is used to determine the value of the most significant bits of the picture order count value of the i-th long-term reference picture that is included in the long-term reference picture set of the current picture.

The variable DeltaPocMSBCycleLt[ i ] is derived as follows.

if( i = = 0 | | poc\_lsb\_lt[ i − 1 ] ! = poc\_lsb\_lt[ i ] )   
 DeltaPocMSBCycleLt[ i ] = delta\_poc\_msb\_cycle\_lt[ i ]   
 else (7‑37)  
 DeltaPocMSBCycleLt[ i ] = delta\_poc\_msb\_cycle\_lt[ i ] + DeltaPocMSBCycleLt[ i − 1 ]

The value of DeltaPocMSBCycleLt[ i ] \* MaxPicOrderCntLsb + pic\_order\_cnt\_lsb – poc\_lsb\_lt[ i ] shall be in the range of 1 to 224 – 1, inclusive.

**used\_by\_curr\_pic\_lt\_flag**[ i ] equal to 0 specifies that the i-th long-term reference picture included in the long-term reference picture set of the current picture is not used for reference by the current picture.

**aps\_pred\_weight\_table\_present\_flag** equal to 1 specifies that weighted prediction tables are present in the APS; equal to 0 specifies that weighted prediction tables are not present in the APS. aps\_pred\_weight\_table\_present\_flag shall be 1 if weighted\_pred\_flag is equal to 1 and slice\_type is P, or weighted\_bipred\_idc is set to 1 and slice\_type is B.

**aps\_adaptive\_loop\_filter\_flag** equal to 1 specifies that the ALF is on for slices referring to the current APS; equal to 0 specifies that the ALF is off for slices referring to the current APS. When there is no active APS, the aps\_adaptive\_loop\_filter\_flag value is inferred to be 0.

**aps\_extension\_flag** equal to 0 specifies that no aps\_extension\_data\_flag syntax elements are present in the picture parameter set RBSP syntax structure. aps\_extension\_flag shall be equal to 0 in bitstreams conforming to this Recommendation | International Standard. The value of 1 for aps\_extension\_flag is reserved for future use by ITU‑T | ISO/IEC. Decoders shall ignore all data that follow the value 1 for aps\_extension\_flag in a picture parameter set NAL unit.

**aps\_extension\_data\_flag** may have any value. Its value does not affect decoder conformance to profiles specified in this Recommendation | International Standard.

Replace 7.3.3 Slice header syntax

With

|  |  |
| --- | --- |
| slice\_header( ) { | Descriptor |
| **first\_slice\_in\_pic\_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( RapPicFlag ) { |  |
| **rap\_pic\_id** | ue(v) |
| **no\_output\_of\_prior\_pics\_flag** | u(1) |
| } |  |
| ~~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 ) {~~ |  |
| **~~num\_long\_term\_pics~~** | ~~ue(v)~~ |
| ~~for( i = 0; i < num\_long\_term\_pics; i++ ) {~~ |  |
| **~~poc\_lsb\_lt~~**~~[ i ]~~ | ~~u(v)~~ |
| **~~delta\_poc\_msb\_present\_flag~~**~~[ i ]~~ | ~~u(1)~~ |
| ~~if( delta\_poc\_msb\_present\_flag[ i ] )~~ |  |
| **~~delta\_poc\_msb\_cycle\_lt~~**~~[ i ]~~ | ~~ue(v)~~ |
| **~~used\_by\_curr\_pic\_lt\_flag~~**~~[ i ]~~ | ~~u(1)~~ |
| ~~}~~ |  |
| ~~}~~ |  |
| ~~}~~ |  |
| if( sample\_adaptive\_offset\_enabled\_flag ) { |  |
| **slice\_sample\_adaptive\_offset\_flag[** 0 **]** | u(1) |
| if( slice\_sample\_adaptive\_offset\_flag[ 0 ] ) { |  |
| **slice\_sample\_adaptive\_offset\_flag[** 1 **]** | u(1) |
| **slice\_sample\_adaptive\_offset\_flag[** 2 **]** | u(1) |
| } |  |
| } |  |
| if(adaptive\_loop\_filter\_enabled\_flag ||  ! IdrPicFlag ||  ( weighted\_pred\_flag && slice\_type == P) | |  ( weighted\_bipred\_idc == 1 && slice\_type == B ) ) |  |
| **aps\_id** | ue(v) |
| if( slice\_type = = P | | slice\_type = = B ) { |  |
| if( sps\_temporal\_mvp\_enable\_flag ) |  |
| **pic\_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 && slice\_type != I ) |  |
| **cabac\_init\_flag** | u(1) |
| **slice\_qp\_delta** | 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( pic\_temporal\_mvp\_enable\_flag ) { |  |
| if( slice\_type = = B ) |  |
| **collocated\_from\_l0\_flag** | u(1) |
| if( slice\_type != I &&   ((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) |
| } |  |
| if( ( weighted\_pred\_flag && slice\_type = = P) | |  ( weighted\_bipred\_idc = = 1 && slice\_type = = B ) ) |  |
| **pred\_weight\_table\_override\_flag** | u(1) |
| if (pred\_weight\_table\_override\_flag) |  |
| pred\_weight\_table( ) |  |
| if( slice\_type = = P | | slice\_type = = B ) |  |
| **five\_minus\_max\_num\_merge\_cand** | ue(v) |
| if( adaptive\_loop\_filter\_enabled\_flag ) { |  |
| **slice\_adaptive\_loop\_filter\_flag** | u(1) |
| if( slice\_adaptive\_loop\_filter\_flag && alf\_coef\_in\_slice\_flag ) |  |
| alf\_param( ) |  |
| if( slice\_adaptive\_loop\_filter\_flag && !alf\_coef\_in\_slice\_flag ) |  |
| alf\_cu\_control\_param( ) |  |
| } |  |
| if( seq\_loop\_filter\_across\_slices\_enabled\_flag &&  ( slice\_adaptive\_loop\_filter\_flag | | slice\_sample\_adaptive\_offset\_flag | |  !disable\_deblocking\_filter\_flag ) ) |  |
| **slice\_loop\_filter\_across\_slices\_enabled\_flag** | u(1) |
| } |  |
| if( tiles\_or\_entropy\_coding\_sync\_idc = = 1 | |  tiles\_or\_entropy\_coding\_sync\_idc = = 2 ) { |  |
| **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** | u(8) |
| } |  |
| byte\_alignment( ) |  |
| } |  |

### Replace 7.4.3 Slice header semantics with

When present, the value of the slice header syntax elements pic\_parameter\_set\_id, pic\_output\_flag, rap\_pic\_id, no\_output\_of\_prior\_pics\_flag, ~~pic\_order\_cnt\_lsb, short\_term\_ref\_pic\_set\_sps\_flag, short\_term\_ref\_pic\_set\_idx, num\_long\_term\_pics and pic\_temporal\_mvp\_enable\_flag~~ shall be the same in all slice headers of a coded picture. ~~When present, the value of the slice header syntax elements poc\_lsb\_lt[ i ], delta\_poc\_msb\_present\_flag[ i ], delta\_poc\_msb\_cycle\_lt[ i ] and used\_by\_curr\_pic\_lt\_flag[ i ] shall be the same in all slice headers of a coded picture for each i in the range of 0 to num\_long\_term\_pics, inclusive.~~

**first\_slice\_in\_pic\_flag** indicates whether the slice is the first slice of the picture. If first\_slice\_in\_pic\_flag is equal to 1, the variables SliceCbAddrZS and SliceCtbAddrRS are both set to 0 and the decoding starts with the first coding tree block in the picture.

**pic\_parameter\_set\_id** specifies the picture parameter set in use. The value of pic\_parameter\_set\_id shall be in the range of 0 to 255, inclusive.

**slice\_address** specifies the address in slice granularity resolution in which the slice starts. The length of the slice\_address syntax element is ( Ceil( Log2( PicWidthInCtbs \* PicHeightInCtbs ) ) + SliceGranularity ) bits.

The variable SliceCtbAddrRS, specifying the coding tree block in which the slice starts in coding tree block raster scan order, is derived as follows.

SliceCtbAddrRS = ( slice\_address >> SliceGranularity ) (7‑34)

The variable SliceCbAddrZS, specifying the address of first coding block in the slice in minimum coding block granularity in z-scan order, is derived as follows.

SliceCbAddrZS = slice\_address << ( ( log2\_diff\_max\_min\_coding\_block\_size − SliceGranularity ) <<1 ) (7‑35)

[Ed. (BB): Change it that SliceCbAddrZS is in z-scan order in a coding tree block while the CTBs are in tiles scan order]

The slice decoding starts with the largest coding unit possible at the slice starting coordinate. [Ed. (BB): More precise description is needed here.]

**dependent\_slice\_flag** equal to 1 specifies that the value of each slice header syntax element not present is inferred to be equal to the value of corresponding slice header syntax element in the preceding slice containing the coding tree block for which the coding tree block address is SliceCtbAddrRS − 1. When not present, the value of dependent\_slice\_flag is inferred to be equal to 0. The value of dependent\_slice\_flag shall be equal to 0 when SliceCtbAddrRS equal to 0.

**slice\_type** specifies the coding type of the slice according to .

Table 7‑7 – Name association to slice\_type

|  |  |
| --- | --- |
| slice\_type | Name of slice\_type |
| 0 | B (B slice) |
| 1 | P (P slice) |
| 2 | I (I slice) |

When nal\_unit\_type is equal to a value in the range of 4 to 8, inclusive (RAP picture), slice\_type shall be equal to 2.

When sps\_max\_dec\_pic\_buffering[ sps\_max\_temporal\_layers\_minus1 ] is equal to 0, slice\_type shall be equal to 2.

**colour\_plane\_id** specifies the colour plane associated with the current slice RBSP when separate\_colour\_plane\_flag is equal to 1. The value of colour\_plane\_id shall be in the range of 0 to 2, inclusive. colour\_plane\_id equal to 0, 1, and 2 correspond to the Y, Cb, and Cr planes, respectively.

NOTE 1 – There is no dependency between the decoding processes of pictures having different values of colour\_plane\_id.

**pic\_output\_flag** affects the decoded picture output and removal processes as specified in Annex C. When pic\_output\_flag is not present, it is inferred to be equal to 1.

**rap\_pic\_id** identifies a RAP picture. The values of rap\_pic\_id in all the slices of a RAP picture shall remain unchanged. When two consecutive access units in decoding order are both RAP access units, the value of rap\_pic\_id in the slices of the first such RAP access unit shall differ from the rap\_pic\_id in the second such RAP access unit. The value of rap\_pic\_id shall be in the range of 0 to 65535, inclusive.

**no\_output\_of\_prior\_pics\_flag** specifies how the previously-decoded pictures in the decoded picture buffer are treated after decoding of an IDR or a BLA picture. See Annex . When the IDR or BLA picture is the first IDR or BLA picture in the bitstream, the value of no\_output\_of\_prior\_pics\_flag has no effect on the decoding process. When the IDR or BLA picture is not the first IDR or BLA picture in the bitstream and the value of pic\_width\_in\_luma\_samples or pic\_height\_in\_luma\_samples or sps\_max\_dec\_pic\_buffering[ sps\_max\_temporal\_layers\_minus1 ] derived from the active sequence parameter set is different from the value of pic\_width\_in\_luma\_samples or pic\_height\_in\_luma\_samples or sps\_max\_dec\_pic\_buffering[ sps\_max\_temporal\_layers\_minus1 ] derived from the sequence parameter set active for the preceding picture, no\_output\_of\_prior\_pics\_flag equal to 1 may (but should not) be inferred by the decoder, regardless of the actual value of no\_output\_of\_prior\_pics\_flag.

**~~pic\_order\_cnt\_lsb~~** ~~specifies the picture order count modulo MaxPicOrderCntLsb for the current picture. The length of the pic\_order\_cnt\_lsb syntax element is log2\_max\_pic\_order\_cnt\_lsb\_minus4 + 4 bits. The value of the pic\_order\_cnt\_lsb shall be in the range of 0 to MaxPicOrderCntLsb − 1, inclusive. When pic\_order\_cnt\_lsb is not present, pic\_order\_cnt\_lsb is inferred to be equal to 0.~~

**~~short\_term\_ref\_pic\_set\_sps\_flag~~** ~~equal to 1 specifies that the short-term reference picture set of the current picture shall be created using syntax elements in the active sequence parameter set. short\_term\_ref\_pic\_set\_sps\_flag equal to 0 specifies that the short-term reference picture set of the current picture shall be created using syntax elements in the short\_term\_ref\_pic\_set( ) syntax structure in the slice header.~~

**~~short\_term\_ref\_pic\_set\_idx~~** ~~specifies the index to the list of the short-term reference picture sets specified in the active sequence parameter set that shall be used for creation of the reference picture set of the current picture. The syntax element short\_term\_ref\_pic\_set\_idx shall be represented by Ceil( Log2( num\_short\_term\_ref\_pic\_sets ) ) bits. The value of short\_term\_ref\_pic\_set\_idx shall be in the range of 0 to num\_short\_term\_ref\_pic\_sets − 1, inclusive, where num\_short\_term\_ref\_pic\_sets is the syntax element from the active sequence parameter set.~~

~~The variable StRpsIdx is derived as follows.~~

~~if( short\_term\_ref\_pic\_set\_sps\_flag )  
 StRpsIdx = short\_term\_ref\_pic\_set\_idx (7‑36)  
else  
 StRpsIdx = num\_short\_term\_ref\_pic\_sets~~

**~~num\_long\_term\_pics~~** ~~specifies the number of the long-term reference pictures that are to be included in the long-term reference picture set of the current picture. The value of num\_long\_term\_pics shall be in the range of 0 to sps\_max\_dec\_pic\_buffering[ sps\_max\_temporal\_layers\_minus1 ] – NumNegativePics[ StRpsIdx ] – NumPositivePics[ StRpsIdx ] , inclusive. When not present, the value of num\_long\_term\_pics is inferred to be equal to 0.~~

**~~poc\_lsb\_lt~~**~~[ i ] specifies the value of the least significant bits of the picture order count value of the i-th long-term reference picture that is included in the long-term reference picture set of the current picture. The length of the poc\_lsb\_lt[ i ] syntax element is log2\_max\_pic\_order\_cnt\_lsb\_minus4 + 4 bits. For any values of j and k in the range of 0 to num\_long\_term\_pics – 1, inclusive, if j is less than k, poc\_lsb\_lt[ j ] shall not be less than poc\_lsb\_lt[ k ].~~

**~~delta\_poc\_msb\_present\_flag[ i ]~~** ~~equal to 1 specifies that delta\_poc\_msb\_cycle\_lt\_minus1[ i ] is present. delta\_poc\_msb\_present\_flag[ i ]~~~~equal to 0 specifies that delta\_poc\_msb\_cycle\_lt\_minus1[ i ] is not present. delta\_poc\_msb\_present\_flag[ i ] shall be equal to 1 when there is more than one reference picture in the decoded picture buffer with the least significant bits of the picture order count value equal to poc\_lsb\_lt[ i ].~~

**~~delta\_poc\_msb\_cycle\_lt~~**~~[ i ]~~~~is used to determine the value of the most significant bits of the picture order count value of the i-th long-term reference picture that is included in the long-term reference picture set of the current picture.~~

~~The variable DeltaPocMSBCycleLt[ i ] is derived as follows.~~

~~if( i = = 0 | | poc\_lsb\_lt[ i − 1 ] ! = poc\_lsb\_lt[ i ] )   
 DeltaPocMSBCycleLt[ i ] = delta\_poc\_msb\_cycle\_lt[ i ]   
 else (7‑37)  
 DeltaPocMSBCycleLt[ i ] = delta\_poc\_msb\_cycle\_lt[ i ] + DeltaPocMSBCycleLt[ i − 1 ]~~

~~The value of DeltaPocMSBCycleLt[ i ] \* MaxPicOrderCntLsb + pic\_order\_cnt\_lsb – poc\_lsb\_lt[ i ] shall be in the range of 1 to 2~~~~24~~~~– 1, inclusive.~~

**~~used\_by\_curr\_pic\_lt\_flag~~**~~[ i ] equal to 0 specifies that the i-th long-term reference picture included in the long-term reference picture set of the current picture is not used for reference by the current picture.~~

**pic\_temporal\_mvp\_enable\_flag** specifies whether temporal motion vector predictors can be used for inter prediction. If pic\_temporal\_mvp\_enable\_flag is equal to 0, the temporal motion vector predictors shall not be used in decoding of the current picture. If pic\_temporal\_mvp\_enable\_flag is equal to 1, temporal motion vector predictors may be used in decoding of the current picture. When not present, the value of pic\_temporal\_mvp\_enable\_flag shall be inferred to be equal to 0.

When both pic\_temporal\_mvp\_enable\_flag and temporal\_id are equal to 0, all coded pictures that follow the current picture in decoding order shall not use temporal motion vector vectors from any picture that precedes the current picture in decoding order.

**num\_ref\_idx\_active\_override\_flag** equal to 1 specifies that the syntax element num\_ref\_idx\_l0\_active\_minus1 is present for P and B slices and that the syntax element num\_ref\_idx\_l1\_active\_minus1 is present for B slices. num\_ref\_idx\_active\_override\_flag equal to 0 specifies that the syntax elements num\_ref\_idx\_l0\_active\_minus1 and num\_ref\_idx\_l1\_active\_minus1 are not present.

**num\_ref\_idx\_l0\_active\_minus1** specifies the maximum reference index for reference picture list 0 that may be used to decode the slice. num\_ref\_idx\_l0\_active\_minus1 shall be in the range of 0 to 15, inclusive. [Ed. (GJS): Constrain to the value imposed by the max DPB size limit instead of 15. (YK): It should be allowed to use the max number of active entries greater than the max DPB size limit as part of the level definition.]

When the current slice is a P or B slice and num\_ref\_idx\_l0\_active\_minus1 is not present, num\_ref\_idx\_l0\_active\_minus1 is inferred to be equal to num\_ref\_idx\_l0\_default\_active\_minus1.

**num\_ref\_idx\_l1\_active\_minus1** specifies the maximum reference index for reference picture list 1 that shall be used to decode the slice.

When the current slice is a B slice and num\_ref\_idx\_l1\_active\_minus1 is not present, num\_ref\_idx\_l1\_active\_minus1 is inferred to be equal to num\_ref\_idx\_l1\_default\_active\_minus1.

The range of num\_ref\_idx\_l1\_active\_minus1 is constrained as specified in the semantics for num\_ref\_idx\_l0\_active\_minus1 with l0 and list 0 replaced by l1 and list 1, respectively.

**mvd\_l1\_zero\_flag** equal to 1 indicates that difference between a list 1 vector component and its prediction, mvd\_l1[][][], is not parsed and it is set equal to 0. mvd\_l1\_zero\_flag equal to 0 indicates that mvd\_l1[][][] is parsed.

**cabac\_init\_flag** specifies the method for determining the initialisation table used in the initialisation process for context variables. The value of cabac\_init\_flag shall be in the range of 0 to 1, inclusive. When cabac\_init\_flag is not present, it is inferred to be 0.

**slice\_qp\_delta** specifies the initial value of QPY to be used for all the coding blocks in the slice until modified by the value of cu\_qp\_delta in the coding unit layer. The initial QPY quantization parameter for the slice is computed as

SliceQPY = 26 + pic\_init\_qp\_minus26 + slice\_qp\_delta (7‑38)

The value of slice\_qp\_delta shall be limited such that SliceQPY is in the range of −QpBdOffsetY to +51, inclusive.

**deblocking\_filter\_override\_flag** equal to 0 specifies that deblocking parameters from the active picture parameter set shall be used for deblocking the current slice. deblocking\_filter\_override\_flag equal to 0 specifies that deblocking parameters from the slice header shall be used for deblocking the current slice. When not present, the value of deblocking\_filter\_override\_flag is inferred to be equal to 1.

**slice\_header\_disable\_deblocking\_filter\_flag** equal to 1 specifies that the operation of the deblocking filter shall not be applied for the current slice. slice\_header\_disable\_deblocking\_filter\_flag equal to 0 specifies that the operation of the deblocking filter shall be applied for the current slice.

**beta\_offset\_div2** and **tc\_offset\_div2** specify the deblocking parameter offsets for β and tC (divided by 2) for the current slice.

**slice\_sample\_adaptive\_offset\_flag[** cIdx **]** equal to 1 specifies that SAO is enabled for the colour component cIdx in the current slice; equal to 0 specifies that SAO is disabled for the colour component cIdx in the current slice. When slice\_sample\_adaptive\_offset\_flag[ cIdx ] is not present, slice\_sample\_adaptive\_offset\_flag[ cIdx ] is inferred to be equal to 0.

**collocated\_from\_l0\_flag** equal to 1 specifies the picture that contains the collocated partition shall be derived from list 0, otherwise the picture shall be derived from list 1.When collocated\_from\_l0\_flag is not present, it is inferred to be equal to 1.

**collocated\_ref\_idx** specifies the reference index of the picture that contains the collocated partition. When the current slice is a P slice collocated\_ref\_idx refers to list 0. When the current slice is a B slice collocated\_ref\_idx refers to list 0 if collocated\_from\_l0 is 1, otherwise it refers to list 1. Collocated\_ref\_idx shall always refer to a valid list entry, and the resulting picture shall be the same for all slices of a coded picture. When collocated\_ref\_idx is not present, it is inferred to be equal to 0.

**pred\_weight\_table\_override\_flag** equal to 1 specifies that weighted prediction tables are present in the slice header and used for the current slice; equal to 0 specified that weighted prediction tables are not present in the slice header, and default weighted prediction tables in the adaptation parameter set are used for the current slice. If pred\_weight\_table\_override\_flag is not present in the bitstream, it is inferred to 0.

**five\_minus\_max\_num\_merge\_cand** specifies the maximum number of merging MVP candidates supported in the slice subtracted from 5. The maximum number of merging MVP candidates, MaxNumMergeCand is computed as

MaxNumMergeCand = 5 − five\_minus\_max\_num\_merge\_cand (7‑39)

The value of five\_minus\_max\_num\_merge\_cand shall be limited such that MaxNumMergeCand is in the range of 0 to 5, inclusive.

**slice\_adaptive\_loop\_filter\_flag** equal to 1 specifies that the ALF is on for the current slice; equal to 0 specifies that the ALF is off for the current slice. It is a requirement of the bitstream that if there is an active APS, the value of slice\_adaptive\_loop\_filter\_flag shall be equal to aps\_adaptive\_loop\_filter\_flag.

**slice\_loop\_filter\_across\_slices\_enabled\_flag** equal to 1 specifies that in-loop filtering operations are performed across slice boundaries; otherwise, the in-loop operations are slice-independent and not applied across slice boundaries. The in-loop filtering operations include the deblocking filter, sample adaptive offset filter, and adaptive loop filter. When slice\_loop\_filter\_across\_slices\_enabled\_flag is not present, it is inferred to be equal to seq\_loop\_filter\_across\_slices\_enabled\_flag.

**num\_entry\_point\_offsets** specifies the number of entry\_point\_offset[ i ] syntax elements in the slice header. When tiles\_or\_entropy\_coding\_sync\_idc is equal to 1, the value of num\_entry\_point\_offsets shall be in the range of 0 to ( num\_tile\_columns\_minus1 + 1 ) \* ( num\_tile\_rows\_minus1 + 1 ) − 1, inclusive. When tiles\_or\_entropy\_coding\_sync\_idc is equal to 2, the value of num\_entry\_point\_offsets shall be in the range of 0 to PicHeightInCtbs − 1, inclusive. When not present, the value of num\_entry\_point\_offsets is inferred to be equal to 0.

**offset\_len\_minus1** plus 1 specifies the length, in bits, of the entry\_point\_offset[ i ] syntax elements.

**entry\_point\_offset**[ i ] specifies the i-th entry point offset, in bytes and shall be represented by offset\_len\_minus1 plus 1 bits. The coded slice data after the slice header consists of num\_entry\_point\_offsets + 1 subsets, with subset index values ranging from 0 to num\_entry\_point\_offsets, inclusive. Subset 0 consists of bytes 0 to entry\_point\_offset[ 0 ] − 1, inclusive, of the coded slice data, subset k, with k in the range of 1 to num\_entry\_point\_offsets - 1, inclusive, consists of bytes entry\_point\_offset[ k − 1 ] to entry\_point\_offset[ k ] + entry\_point\_offset[ k − 1 ] − 1, inclusive, of the coded slice data, and the last subset (with subset index equal to num\_entry\_point\_offsets) consists of the remaining bytes of the coded slice data.

When tiles\_or\_entropy\_coding\_sync\_idc is equal to 1 and num\_entry\_point\_offsets is greater than 0, each subset shall contain all coded bits of exactly one tile, and the number of subsets (i.e., the value of num\_entry\_point\_offsets + 1) shall be equal to or less than the number of tiles in the slice.

NOTE 2 – When tiles\_or\_entropy\_coding\_sync\_idc is equal to 1, each slice must include either a subset of one tile (in which case signalling of entry points is unnecessary) or an integer number of complete tiles.

When tiles\_or\_entropy\_coding\_sync\_idc is equal to 2 and num\_entry\_point\_offsets is greater than 0, each subset k with k in the range of 0 to num\_entry\_point\_offsets − 1, inclusive, shall contain all coded bits of exactly one row of coding tree blocks, the last subset (with subset index equal to num\_entry\_point\_offsets) shall contain all coded bits of the remaining coding blocks included in the slice, wherein the remaining coding blocks consist of either exactly one row of coding tree blocks or a subset of one row of coding tree blocks, and the number of subsets (i.e., the value of num\_entry\_point\_offsets + 1) shall be equal to the number of rows of coding tree blocks in the slice, wherein a subset of one row of coding tree blocks in the slice is also counted..

NOTE 3 – When tiles\_or\_entropy\_coding\_sync\_idc is equal to 2, a slice may include a number of rows of coding tree blocks and a subset of a row of coding tree blocks. For example, if a slice include two and a half rows of coding tree blocks, the the number of subsets (i.e., the value of num\_entry\_point\_offsets + 1) shall be equal to 3.

**slice\_header\_extension\_length** specifies the length of the slice header extension data in bytes, not including the bits used for signalling slice\_header\_extension\_length itself. The value of slice\_header\_extension\_length shall be in the range of 0 to 256, inclusive.

**slice\_header\_extension\_data\_byte** may have any value. Decoders shall ignore the value of slice\_header\_extension\_data\_byte. Its value does not affect decoder conformance to profiles specified in this Recommendation | International Standard.

#### Replace 7.3.3.7 Prediction weight table syntax with

|  |  |
| --- | --- |
| pred\_weight\_table( ) { | Descriptor |
| **luma\_log2\_weight\_denom** | ue(v) |
| if( chroma\_format\_idc != 0 ) |  |
| **delta\_chroma\_log2\_weight\_denom** | se(v) |
| ~~if( slice\_type = = P | | slice\_type = = B )~~ |  |
| **list0\_pred\_weight\_table\_present\_flag** | u(1) |
| If (list0\_pred\_weight\_table\_present\_flag) |  |
| for( i = 0; i <= num\_ref\_idx\_l0\_active\_minus1; i++ ) { |  |
| **luma\_weight\_l0\_flag** | u(1) |
| if( luma\_weight\_l0\_flag ) { |  |
| **delta\_luma\_weight\_l0[** i **]** | se(v) |
| **luma\_offset\_l0[** i **]** | se(v) |
| } |  |
| if( chroma\_format\_idc != 0 ) { |  |
| **chroma\_weight\_l0\_flag** | u(1) |
| if( chroma\_weight\_l0\_flag ) |  |
| for( j =0; j < 2; j++ ) { |  |
| **delta\_chroma\_weight\_l0[** i **][** j **]** | se(v) |
| **delta\_chroma\_offset\_l0[** i **][** j **]** | se(v) |
| } |  |
| } |  |
| } |  |
| ~~if( slice\_type = = B )~~ |  |
| **list1\_pred\_weight\_table\_present\_flag** | u(1) |
| If (list1\_pred\_weight\_table\_present\_flag) |  |
| for( i = 0; i <= num\_ref\_idx\_l1\_active\_minus1; i++ ) { |  |
| **luma\_weight\_l1\_flag** | u(1) |
| if( luma\_weight\_l1\_flag ) { |  |
| **delta\_luma\_weight\_l1[** i **]** | se(v) |
| **luma\_offset\_l1[** i **]** | se(v) |
| } |  |
| if( chroma\_format\_idc != 0 ) { |  |
| **chroma\_weight\_l1\_flag** | u(1) |
| if( chroma\_weight\_l1\_flag ) |  |
| for( j = 0; j < 2; j++ ) { |  |
| **delta\_chroma\_weight\_l1[** i **][** j **]** | se(v) |
| **delta\_chroma\_offset\_l1[** i **][** j **]** | se(v) |
| } |  |
| } |  |
| } |  |
| } |  |

#### Replace 7.4.3.7 Weighted prediction parameters semantics with

**luma\_log2\_weight\_denom** is the base 2 logarithm of the denominator for all luma weighting factors. The value of luma\_log2\_weight\_denom shall be in the range of 0 to 7, inclusive.

**delta\_chroma\_log2\_weight\_denom** is the difference of the base 2 logarithm of the denominator for all chroma weighting factors.

The variable ChromaLog2WeightDenom is specified by luma\_log2\_weight\_denom + delta\_chroma\_log2\_weight\_denom and it shall be in the range of 0 to 7, inclusive.

**list0\_pred\_weight\_table\_present\_flag equal to 1** specifies that weighting factors for list 0 prediction are present. list0\_pred\_weight\_table\_present\_flag equal to 0 specifies that these weighting factors are not present. For weighted prediction parameters carried in APS, list0\_pred\_weight\_table\_present\_flag shall set equal to 1 if any of slices in the current picture is a P- or B-slice; For weighted prediction parameters carried in slice header, list0\_pred\_weight\_table\_present\_flag shall set equal to 1 if the current slice is a P- or B-slice;

**luma\_weight\_l0\_flag** equal to 1 specifies that weighting factors for the luma component of list 0 prediction are present. luma\_weight\_l0\_flag equal to 0 specifies that these weighting factors are not present.

**delta\_luma\_weight\_l0[** i **]** is the difference of the weighting factor applied to the luma prediction value for list 0 prediction using RefPicList0[ i ].

The variable LumaWeightL0[ i ] is specified by (1 << luma\_log2\_weight\_denom ) + delta\_luma\_weight\_l0[ i ]. When luma\_weight\_l0\_flag is equal to 1, the value of LumaWeightL0[ i ] shall be in the range of −128 to 127, inclusive. When luma\_weight\_l0\_flagis equal to 0, LumaWeightL0[ i ] is inferred to be equal to 2luma\_log2\_weight\_denom for RefPicList0[ i ].

**luma\_offset\_l0[** i **]** is the additive offset applied to the luma prediction value for list 0 prediction using RefPicList0[ i ]. The value of luma\_offset\_l0[ i ] shall be in the range of −128 to 127, inclusive. When luma\_weight\_l0\_flagis equal to 0, luma\_offset\_l0[ i ] is inferred as equal to 0 for RefPicList0[ i ].

**chroma\_weight\_l0\_flag** equal to 1 specifies that weighting factors for the chroma prediction values of list 0 prediction are present. chroma\_weight\_l0\_flag equal to 0 specifies that these weighting factors are not present.

**delta\_chroma\_weight\_l0[** i **][** j **]** is the difference of the weighting factor applied to the chroma prediction values for list 0 prediction using RefPicList0[ i ] with j equal to 0 for Cb and j equal to 1 for Cr.

The variable ChromaWeightL0[ i ][ j ] is specified by ( 1 << ChromaLog2WeightDenom ) + delta\_chroma\_weight\_l0[ i ][ j ]. When chroma\_weight\_l0\_flag is equal to 1, the value of ChromaWeightL0[ i ][ j ] shall be in the range of −128 to 127, inclusive. When chroma\_weight\_l0\_flag is equal to 0**,** ChromaWeightL0[ i ][ j ] is inferred to be equal to 2ChromaLog2WeightDenom for RefPicList0[ i ].

**delta\_chroma\_offset\_l0[** i **][** j **]** is the difference of the additive offset applied to the chroma prediction values for list 0 prediction using RefPicList0[ i ] with j equal to 0 for Cb and j equal to 1 for Cr.

The variable ChromaOffsetL0[ i ][ j ] is specified as follows:

shift = 1 << ( BitDepthC − 1 )

ChromaOffsetL0[ i ][ j ] = (delta\_chroma\_offset\_l0[i][j] –   
 ( (shift\*ChromaWeightL0[ i ][ j ]) >> ChromaLog2WeightDenom ) − shift ) (7‑65)

The variable ChromaOffsetL0[ i ][ j ] shall be in the range of −127 to 128, inclusive. When chroma\_weight\_l0\_flag is equal to 0**,** ChromaOffsetL0[ i ][ j ] is inferred to be equal to 0 for RefPicList0[ i ].

**list1\_pred\_weight\_table\_present\_flag equal to 1** specifies that weighting factors for list 1 prediction are present. List1\_pred\_weight\_table\_present\_flag equal to 0 specifies that these weighting factors are not present. For weighted prediction parameters carried in APS, list1\_pred\_weight\_table\_present\_flag shall set equal to 1 if any of slices in the current picture is a B-slice; For weighted prediction parameters carried in slice header, list0\_pred\_weight\_table\_present\_flag shall set equal to 1 if the current slice is a B-slice;

**luma\_weight\_l1\_flag, delta\_luma\_weight\_l1**, **luma\_offset\_l1**, **chroma\_weight\_l1\_flag**, **delta\_chroma\_weight\_l1**, **delta\_chroma\_offset\_l1** have the same semantics as luma\_weight\_l0\_flag, delta\_luma\_weight\_l0, luma\_offset\_l0, chroma\_weight\_l0\_flag, delta\_chroma\_weight\_l0, delta\_chroma\_offset\_l0, respectively, with l0, list 0, and List0 replaced by l1, list 1, and List1, respectively.