|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  10th Meeting: Stockholm, SE, 11–20 July 2012 | Document: JCTVC-J0115 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **AHG13: Signalling of long-term reference pictures in the slice header** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Adarsh K. Ramasubramonian Ye-Kui Wang Rajan Joshi Ying Chen  5775 Morehouse Drive, San Diego, CA 92121-1714 USA | Tel: Email: | 1-858-658-5804 [aramasub@qualcomm.com](mailto:aramasub@qualcomm.com)  1-858-651-8345 [yekuiw@qualcomm.com](mailto:yekuiw@qualcomm.com)  1-858-658-4511 [rajanj@qualcomm.com](mailto:rajanj@qualcomm.com)  1-858-845-6589 [cheny@qualcomm.com](mailto:cheny@qualcomm.com) |
| *Source:* | Qualcomm Incorporated | | |

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

# Abstract

An error-resilience problem is reported for the current signalling and derivation of long-term reference pictures (LTRPs). It is stated that, the root of problem was that the reference picture set (RPS) derivation depends on the status of the decoded picture buffer (DPB), which was the main reason for replacing the sliding-window and memory management control operation (MMCO) based reference picture buffer management mechanism in AVC with the RPS based mechanism. This document proposes to send the delta POC values, either between the current picture and the LTRP or between the LTRP and the previous random access point (RAP) picture in decoding order to solve the reported error resiliency issue, and to satisfy the RPS design principle that the RPS derivation is self-contained (i.e., not depending on the DPB status). The simulation results are in the attachment of the proposal document.

# Introduction

This document firstly discusses the problems in the slice header signalling of LTRPs in the presence of picture losses. The overall assumption is that the RPS derivation must be robust to picture losses, as long as relative POC values can be correctly established, for both STRPs and LTRPs. This requires that the RPS derivation for any picture be self-contained, i.e., not depending on the DPB status. Such a design principle has actually been the main motivation for replacing the sliding-window and memory management control operation (MMCO) based reference picture buffer management mechanism in AVC with the RPS based mechanism, as described in JCTVC-F493, wherein a different term, namely buffer description, was used for RPS.

With this design principle, the short-term reference pictures (STRPs) are signalled using their delta POC values, with respect to the current picture. Hence, the decoder is able to derive the POC value of these reference pictures and hence check whether each of the STRPs is present. However, in the case of the current signalling and RPS derivation for LTRPs, erroneous derivation of RPS is possible in the presence of losses. This is due to signalling only the amount of POC LSBs for the LTRPs same as for signalling of the POC value for each current picture (i.e., pic\_order\_cnt\_lsb), though in some cases the full POC values using the delta\_msb\_cycle\_lt syntax element are signalled for the LTRPs. If there is a picture in the DPB which has the same LSB as signalled, then there could be ambiguity in the derivation of RPS, and incorrect result of RPS could occur, as shown in the examples presented below. In all the examples, we have MaxPicOrderCntLsb equal to 256 or in other words log2\_max\_pic\_order\_cnt\_lsb\_minus4 equals 4.

## Example 1

Consider three pictures with POC values 383, 384, and 385 that are successive in decoding order. Assume that picture 383 has pictures with POC 0 and 256 as LTRPs, and pictures 384 and 385 only have picture with POC 256 as the LTRP. None of the pictures 383, 384, and 385 have LSB value equal to 0, and we assume than no picture in the DPB has LSB value equal to 0, except pictures 0 and 256. Under the current signalling, the relevant syntax elements of a conforming bitsream are as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| POC (in decoding order) | LTRPs | LTRP LSBs | msb\_present\_flag | msb\_cycle\_lt |
| … | … | … | … | … |
| 383 | 0, 256 | 0, 0 | 1, 1 | 0, 1 |
| 384 | 256 | 0 | 1 | 0 |
| 385 | 256 | 0 | 0 | - |
| … | … | … | … | … |

For picture 384, even though only picture 256 is signalled as the LTRP, when the RPS of picture 384 is derived, picture 0 would still be in the DPB. Hence even for picture 384, msb\_present\_flag is set to 1, and the msb\_cycle value is sent. After RPS derivation for picture 384, picture 0 is marked as “unused for reference” and may be removed from the DPB.

For picture 385, however, since only one LTRP is present in the DPB, msb\_present\_flag is set to 0 and msb\_cycle is not present.

Under the current syntax structure, the above table will correspond to a conforming bitstream. However, if picture 384 is lost, then the LTRP signalling for 385 would be insufficient to resolve which picture from the DPB is to be picked as the LTRP. Both 0 and 256 have LSB of 0, and the decoder would not be able to resolve the issue. The decoder would need the msb\_present flag to be equal to 1 for picture 385 to uniquely identify the picture to be used as the only LTRP.

## Example 2

A related situation is as follows. The description of the table is similar as in example 1, with the exception that picture 0 was signalled as LTRP and picture 256 was signalled as STRP for picture 383.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| POC (in decoding order) | LTRPs | STRPs | LTRP LSBs | msb\_present\_flag | msb\_cycle\_lt |
| … | … | … | … | … | … |
| 383 | 0 | 256 | 0 | 1 | 1 |
| 384 | 256 | - | 0 | 1 | 0 |
| 385 | 256 | - | 0 | 0 | - |
| … | … | … | … | … | … |

Here again, if picture 384 is lost, the decoder would not be able to correctly identify that the LTRP signalled is picture 256, and it would need the msb\_present\_flag to be equal to 1 to uniquely identify the picture to be used as the only LTRP.

## Example 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| POC (in decoding order) | LTRPs | STRP | LTRP LSBs | msb\_present\_flag | msb\_cycle\_lt |
| … | … | … | … | … | … |
| 255 | 0 | - | 0 | 0 |  |
| 256 | 0 | - | 0 | 0 |  |
| 257 | 256 | - | 0 | 1 | 0 |
| 258 | 256 | - | 0 | 0 | - |
| … | … | … | … | … | … |

As shown in the above table, say that picture with POC 255 is received followed by picture 258 (pictures 256 and 257 were lost). After decoding slice header for picture 258, picture 0 (instead of picture 256) will be marked as LTRP. Rather than figuring out that the reference picture 256 is not present for picture 258, the decoder would incorrectly derive that picture 0 is the LTRP signalled and continue decoding (provided other aspects of the decoding process work for this loss).

## Discussion

From the above examples, it may seem that an obvious solution to solve the above problems is to signal the actual full POC value of the LTRPs. It would also seem that the RPS derivation would be self-contained with full POC signalling. However, the possibility that CRA pictures may either start a bitstream, or that the CRA pictures may be converted into a BLA picture by a splicer rules out signalling the full POC value. For example, consider a CRA picture in a bitstream with a POC value of 256 (still assuming MaxPicOrderCntLsb = 256), and let some pictures that follow the CRA picture in decoding order have the CRA picture as an LTRP. If we signal the full POC of the CRA picture, which is 256, and the CRA picture is converted into a BLA picture or starts a bitstream, then the LTRP would be incorrectly derived, and the bitstream will become non-conforming. When a CRA picture is converted to a BLA picture or starts a bitstream, the decoder derives its POC value only on the basis of its LSB; in our example, the POC of the CRA picture (which is now a BLA picture) will be determined to be 0. When pictures look for LTRP with POC 256, the DPB will not contain any such picture because now the CRA picture has derived POC of 0. This would result in a “no reference picture” in the RPS, and if current picture is not a TFD picture associated with the CRA picture, the resulting bitstream would be non-conforming.

Another possible solution is simply mandate the presence of POC MSB cycle, such that the delta of the POC values between the current picture and the LTRP is always signalled. However, when an LTRP is far before the current picture, e.g., when the background of an empty video conferencing room is encoded as the first picture in the bitstream and always used as an LTRP for efficient coding of uncovered background, the delta value can be very large and need a lot of bits to represent.

# Proposal

A solution is proposed to solve the error resilience problem and to make sure that RPS deviation is self-contained (i.e., not depending on the DPB status), and is at the same time problem-free when a random access is performed from a CRA picture or when a CRA picture is converted to a BLA picture. The information signalled for a particular LTRP to be included in the RPS of the current picture is the delta POC value, either between the current picture and the LTRP or between the LTRP and the previous RAP picture in decoding order.

The proposed syntax, semantics and decoding process changes are provided below, wherein additions are highlighted in this color, and removals are strikethrough and highlighted in this color.

## Syntax and semantics

|  |  |  |
| --- | --- | --- |
| slice\_header( ) { | Descriptor | |
| ... |  | |
| 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\_lt\_curr\_pic\_flag**[ i ] | | u(1) |
| **~~delta\_poc\_msb\_present\_flag~~**~~[ i ]~~ | | ~~u(1)~~ |
| ~~if( delta\_poc\_msb\_present\_flag[ i ] )~~ | |  |
| **delta\_poc\_msb\_cycle\_lt**[ i ] | | ue(v) |
| **used\_by\_curr\_pic\_lt\_flag**[ i ] | | u(1) |
| } | |  |
| } | |  |
| ... | |  |

**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\_lt\_curr\_pic\_flag**[ i ] (together with 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.

**~~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 ] (together with delta\_poc\_lt\_curr\_pic\_flag[ 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 PocLt[ i ]~~DeltaPocMSBCycleLt[ i ]~~ is derived as follows.

deltaPocMSBCycleLt[ i ] = delta\_poc\_msb\_cycle\_lt[ i ]  
 if( i != 0 )  
 deltaPocMSBCycleLt[ i ] += deltaPocMSBCycleLt[ i − 1 ]  
 if( delta\_poc\_lt\_curr\_pic\_flag[ i ] )  
 PocLt[ i ] = PicOrderCntMsb – deltaPocMSBCycleLt[ i ] \* MaxPicOrderCntLsb + poc\_lsb\_lt[ i ]  
 else  
 PocLt[ i ] = PrevRapPicPocMsb + deltaPocMSBCycleLt[ i ] \* MaxPicOrderCntLsb + poc\_lsb\_lt[ i ]

Wherein PicOrderCntMsb is the difference between the picture order count value of the current picture and pic\_order\_cnt\_lsb of the current picture, and PrevRapPicPocMsb is difference between the PicOrderCntVal value of the previous RAP picture in decoding order and pic\_order\_cnt\_lsb of the previous RAP picture in decoding order.

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

The value of Abs( PicOrderCntVal –PocLt[ i ] ) ~~DeltaPocMSBCycleLt[ i ] \* MaxPicOrderCntLsb + pic\_order\_cnt\_ls – poc\_lsb\_lt[ i ]~~ shall be in the range of 1 to 224 – 1, inclusive, wherein PicOrderCntVal is the picture order count value of the current picture.

## Decoding process for reference picture set

...

* If the current picture is a CRA picture that is the first coded picture in the bitstream, an IDR picture or a BLA picture, PocStCurrBefore, PocStCurrAfter, PocStFoll, PocLtCurr, and PocLtFoll are all set to empty, and NumPocStCurrBefore, NumPocStCurrAfter, NumPocStFoll, NumPocLtCurr, and NumPocLtFoll are all set to 0.

...

for( i = 0, j = 0, k = 0; i < num\_long\_term\_pics; i++ )  
 ~~if( delta\_poc\_msb\_present\_flag[ i ] )  
 if( used\_by\_curr\_pic\_lt\_flag[ i ] )  
 PocLtCurr[ j++ ] = PicOrderCntVal − DeltaPocMSBCycleLt[ i ] \* MaxPicOrderCntLsb −  
 pic\_order\_cnt\_lsb + poc\_lsb\_lt[ i ]  
 else  
 PocLtFoll[ k++ ] = PicOrderCntVal − DeltaPocMSBCycleLt[ i ] \* MaxPicOrderCntLsb −  
 pic\_order\_cnt\_lsb + poc\_lsb\_lt[ i ]  
 else~~ if( used\_by\_curr\_pic\_lt\_flag[ i ] )  
 PocLtCurr[ j++ ] = PocLt[ i ]~~poc\_lsb\_lt[ i ]~~  
 else  
 PocLtFoll[ k++ ] = PocLt[ i ]~~poc\_lsb\_lt[ i ]~~  
NumPocLtCurr = j  
NumPocLtFoll = k

...

1. The following applies:

for( i = 0; i < NumPocLtCurr; i++ ) {  
 ~~if( !delta\_poc\_msb\_present\_flag[ i ] ) {  
 if( there is a long-term reference picture picX in the DPB [Ed. (JB): Should be made more precise.]  
 with pic\_order\_cnt\_lsb equal to PocLtCurr[ i ] )  
 RefPicSetLtCurr[ i ] = picX  
 else if( there is a short-term reference picture picY in the DPB  
 with pic\_order\_cnt\_lsb equal to PocLtCurr[ i ] )  
 RefPicSetLtCurr[ i ] = picY  
 else   
 RefPicSetLtCurr[ i ] = "no reference picture"  
 } else {~~ if( there is a long-term reference picture picX in the DPB  
 with PicOrderCntVal equal to PocLtCurr[ i ] )  
 RefPicSetLtCurr[ i ] = picX  
 else if( there is a short-term reference picture picY in the DPB  
 with PicOrderCntVal equal to PocLtCurr[ i ] )  
 RefPicSetLtCurr[ i ] = picY  
 else   
 RefPicSetLtCurr[ i ] = "no reference picture"  
 ~~}~~}

for( i = 0; i < NumPocLtFoll; i++ ) {  
 ~~if( !delta\_poc\_msb\_present\_flag[ i ] ) {  
 if( there is a long-term reference picture picX in the DPB  
 with pic\_order\_cnt\_lsb equal to PocLtFoll[ i ] )  
 RefPicSetLtFoll[ i ] = picX  
 else if( there is a short-term reference picture picY in the DPB  
 with pic\_order\_cnt\_lsb equal to PocLtFoll[ i ] )  
 RefPicSetLtFoll[ i ] = picY  
 else   
 RefPicSetLtFoll[ i ] = "no reference picture"  
 } else {~~ if( there is a long-term reference picture picX in the DPB  
 with PicOrderCntVal to PocLtFoll[ i ] )  
 RefPicSetLtFoll[ i ] = picX  
 else if( there is a short-term reference picture picY in the DPB  
 with PicOrderCntVal equal to PocLtFoll[ i ] )  
 RefPicSetLtFoll[ i ] = picY  
 else  
 RefPicSetLtFoll[ i ] = "no reference picture"  
 ~~}~~}

1. All reference pictures included in RefPicSetLtCurr and RefPicSetLtFoll are marked as "used for long-term reference"
2. The following applies:

for( i = 0; i < NumPocStCurrBefore; i++ )  
 if( there is a short-termreference picture picX in the DPB  
 with PicOrderCntVal equal to PocStCurrBefore[ i ])  
 RefPicSetStCurrBefore[ i ] = picX  
 else  
 RefPicSetStCurrBefore[ i ] = "no reference picture"

for( i = 0; i < NumPocStCurrAfter; i++ )  
 if( there is a short-term reference picture picX in the DPB  
 with PicOrderCntVal equal to PocStCurrAfter[ i ])  
 RefPicSetStCurrAfter[ i ] = picX  
 else  
 RefPicSetStCurrAfter[ i ] = "no reference picture" (8‑7)

for( i = 0; i < NumPocStFoll; i++ )  
 if( there is a short-term reference picture picX in the DPB  
 with PicOrderCntVal equal to PocStFoll[ i ])  
 RefPicSetStFoll[ i ] = picX  
 else  
 RefPicSetStFoll[ i ] = "no reference picture"

1. All reference pictures included in RefPicSetStCurrBefore, RefPicSetStCurrAfter and RefPicSetStFoll are marked as "used for short-term reference".
2. All reference pictures in the decoded picture buffer that are not included in RefPicSetLtCurr, RefPicSetLtFoll, RefPicSetStCurrBefore, RefPicSetStCurrAfter or RefPicSetStFoll are marked as "unused for reference".

# Discussion

This section describes how the proposed syntax solves the problems listed in Section 1. For now, it is assumed that in all the three examples, picture with POC 0 is a RAP IDR picture, and no other picture in the bitstreams is a RAP.

## Example 1

With the new syntax, we see that the required numbers of bits for signalling the delta POC values as well as the delta POC values themselves of the LTRPs are in Table 1, for different values of delta\_poc\_lt\_curr\_pic\_flag. Now, even if picture with POC 384 is lost, the RPS derivation for picture 385 would not be affected and for picture 385, it will be correctly decoded that picture 256 is the LTRP.

Table 1: Improved signalling is resilient to one-picture loss in Example 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| POC (in decoding order) | LTRPs | delta\_poc\_lt\_curr\_pic\_flag[ i ] = 0 | | delta\_poc\_lt\_curr\_pic\_flag[ i ] = 1 | |
| poc\_lsb\_lt[ i ] | msb\_cycle\_lt[ i ] | poc\_lsb\_lt[ i ] | msb\_cycle\_lt[ i ] |
| … | … | … |  |  |  |
| 383 | 0, 256 | 0, 0 | 0, 1 | 0, 0 | 0, 1 |
| 384 | 256 | 0 | 1 | 0 | 0 |
| 385 | 256 | 0 | 1 | 0 | 0 |
| … | … | … |  |  |  |

## Example 2

With the new syntax, we see that the required numbers of bits for signalling the delta POC values as well as the delta POC values themselves of the LTRPs are given in Table 2. Similarly as shown for example 1, even if picture with POC 384 is lost, the LTRP signalled for picture 385 would be correctly decoded as picture 256.

Table 2: Improved signalling is resilient to one picture loss in Example 2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| POC (in decoding order) | LTRPs | STRPs | delta\_poc\_lt\_curr\_pic\_flag[ i ] = 0 | | delta\_poc\_lt\_curr\_pic\_flag[ i ] = 1 | |
|  | poc\_lsb\_lt[ i ] | msb\_cycle\_lt[ i ] | poc\_lsb\_lt[ i ] | msb\_cycle\_lt[ i ] |
| … | … |  | … |  |  |  |
| 383 | 0 | 256 | 0 | 0 | 0 | 1 |
| 384 | 256 |  | 0 | 1 | 0 | 0 |
| 385 | 256 |  | 0 | 1 | 0 | 0 |
| … | … |  | … |  |  |  |

## Example 3

The third example in Section 1 considered an example with two-picture loss, where the LTRP signalling in WD7 would result in erroneous RPS. In Table 3, it is shown that even in this case the proposed LTRP signalling would result in the desired RPS derivation. If pictures 256 and 257 are lost, the RPS derivation for picture 258 would correctly identify that the signalled LTRP 256 is not present in the DPB.

Table 3: Improved signalling is resilient to two picture losses in Example 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| POC (in decoding order) | LTRPs | delta\_poc\_lt\_curr\_pic\_flag[ i ] = 0 | | delta\_poc\_lt\_curr\_pic\_flag[ i ] = 1 | |
| poc\_lsb\_lt[ i ] | msb\_cycle\_lt[ i ] | poc\_lsb\_lt[ i ] | msb\_cycle\_lt[ i ] |
| … | … | … |  |  |  |
| 255 | 0 | 0 | 0 | 0 | 0 |
| 256 | 0 | 0 | 0 | 0 | 1 |
| 257 | 256 | 0 | 1 | 0 | 0 |
| 258 | 256 | 0 | 1 | 0 | 0 |
| … | … | … |  |  |  |

# Relationship to JCTVC-J0116

If the proposal of LTRP signaling in SPS is adopted, this proposal can be simply added on top of it. The combined syntax, semantics and decoding process changes are provided below. Additional changes compared to those in section 2 are highlighted in yellow.

## Syntax and semantics

|  |  |
| --- | --- |
| seq\_parameter\_set\_rbsp( ) { | Descriptor |
| **...** |  |
| **num\_short\_term\_ref\_pic\_sets** | ue(v) |
| for( i = 0; i < num\_short\_term\_ref\_pic\_sets; i++) |  |
| short\_term\_ref\_pic\_set( i ) |  |
| **long\_term\_ref\_pics\_present\_flag** | u(1) |
| if( long\_term\_ref\_pics\_present\_flag ) { |  |
| **num\_long\_term\_ref\_pics\_sps** | ue(v) |
| for( i = 0; i < num\_long\_term\_ref\_pics\_sps; i++ ) |  |
| **lt\_ref\_pic\_poc\_lsb\_sps**[ i ] | u(v) |
| } |  |
| ... |  |
| } |  |

**num\_long\_term\_ref\_pics\_sps** specifies the number of long-term reference pictures that are specified in the sequence parameter set. The value of num\_long\_term\_ref\_pics\_sps shall be in the range of 0 to 32, inclusive.

**lt\_ref\_pic\_poc\_lsb\_sps**[ i ] specifies the least significant bits of the picture order count of the i-th long-term reference picture specified in the sequence parameter set. The number of bits used to represent lt\_ref\_pic\_poc\_lsb\_sps[ i ] shall be equal to log2\_max\_pic\_order\_cnt\_lsb\_minus4 + 4.

|  |  |  |
| --- | --- | --- |
| slice\_header( ) { | Descriptor | |
| ... |  | |
| if( long\_term\_ref\_pics\_present\_flag ) { |  | |
| **num\_long\_term\_pics** | ue(v) | |
| if( num\_long\_term\_ref\_pics\_sps ) |  | |
| **num\_long\_term\_sps** | ue(v) | |
| for( i = 0; i < num\_long\_term\_sps + num\_long\_term\_pics; i++ ) { |  | |
| if ( i < num\_long\_term\_sps ) |  | |
| **lt\_idx\_sps**[ i ] | u(v) | |
| else |  | |
| **poc\_lsb\_lt**[ i ] | u(v) | |
| **delta\_poc\_lt\_curr\_pic\_flag**[ i ] | | u(1) |
| **~~delta\_poc\_msb\_present\_flag~~**~~[ i ]~~ | | ~~u(1)~~ |
| ~~if( delta\_poc\_msb\_present\_flag[ i ] )~~ | |  |
| **delta\_poc\_msb\_cycle\_lt**[ i ] | ue(v) | |
| **used\_by\_curr\_pic\_lt\_flag**[ i ] | u(1) | |
| } |  | |
| } |  | |
| ... |  | |

**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 and that are directly signalled in the slice header. 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 ] – num\_long\_term\_sps, inclusive. When not present, the value of num\_long\_term\_pics is inferred to be equal to 0.

**num\_long\_term\_sps** specifies the number of long-term reference pictures that are specified in the active sequence parameter set and that are to be included in the long-term reference picture set of the current picture. If num\_long\_term\_sps is not present, the value is inferred to be equal to 0. The value of num\_long\_term\_sps shall be in the range of 0 to Min( num\_long\_term\_ref\_pics\_sps, max\_dec\_pic\_buffering[ max\_temporal\_layers\_minus1 ] – NumNegativePics[ StRpsIdx ] – NumPositivePics[ StRpsIdx ] – num\_long\_term\_pics ), inclusive.

**lt\_idx\_sps**[ i ] specifies the index, to the list of long-term reference pictures specified in the active sequence parameter set, of the i-th long-term reference picture inherited from the referred sequence parameter set to the long-term reference picture set of the current picture. The value of lt\_idx\_sps[ i ] shall be in the range of 0 to num\_long\_term\_ref\_pics\_sps − 1, inclusive.

**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 ].~~

The variable PocLsbLt[ i ] is derived as follows.

if( i < num\_long\_term\_sps )  
 PocLsbLt[ i ] = lt\_ref\_pic\_poc\_lsb\_sps[ lt\_idx\_sps[ i ] ]  
 else  
 PocLsbLt [ i ] = poc\_lsb\_lt[ i ]

**delta\_poc\_lt\_curr\_pic\_flag**[ i ] (together with 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.

**~~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 ] (together with delta\_poc\_lt\_curr\_pic\_flag[ 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 PocLt[ i ]~~DeltaPocMSBCycleLt[ i ]~~ is derived as follows.

deltaPocMSBCycleLt[ i ] = delta\_poc\_msb\_cycle\_lt[ i ]  
 if( i != 0 | | i != num\_long\_term\_sps )  
 deltaPocMSBCycleLt[ i ] += deltaPocMSBCycleLt[ i − 1 ]  
 if( delta\_poc\_lt\_curr\_pic\_flag[ i ] )  
 PocLt[ i ] = PicOrderCntMsb – deltaPocMSBCycleLt[ i ] \* MaxPicOrderCntLsb + PocLsbLt[ i ]  
 else  
 PocLt[ i ] = PrevRapPicPocMsb + deltaPocMSBCycleLt[ i ] \* MaxPicOrderCntLsb + PocLsbLt[ i ]

Wherein PicOrderCntMsb is the difference between the picture order count value of the current picture and pic\_order\_cnt\_lsb of the current picture, and PrevRapPicPocMsb is difference between the PicOrderCntVal value of the previous RAP picture in decoding order and pic\_order\_cnt\_lsb of the previous RAP picture in decoding order.

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

The value of Abs( PicOrderCntVal –PocLt[ i ] ) ~~DeltaPocMSBCycleLt[ i ] \* MaxPicOrderCntLsb + pic\_order\_cnt\_ls – poc\_lsb\_lt[ i ]~~ shall be in the range of 1 to 224 – 1, inclusive, wherein PicOrderCntVal is the picture order count value of the current picture.

## Decoding process for reference picture set

The only change compared to the RPS decoding process in subsection 2.2 is the replacement of "num\_long\_term\_pics" with "num\_long\_term\_pics + num\_long\_term\_pics", as highlighted below in yellow.

...

for( i = 0, j = 0, k = 0; i < num\_long\_term\_pics + num\_long\_term\_pics; i++ )  
 ~~if( delta\_poc\_msb\_present\_flag[ i ] )  
 if( used\_by\_curr\_pic\_lt\_flag[ i ] )  
 PocLtCurr[ j++ ] = PicOrderCntVal − DeltaPocMSBCycleLt[ i ] \* MaxPicOrderCntLsb −  
 pic\_order\_cnt\_lsb + poc\_lsb\_lt[ i ]  
 else  
 PocLtFoll[ k++ ] = PicOrderCntVal − DeltaPocMSBCycleLt[ i ] \* MaxPicOrderCntLsb −  
 pic\_order\_cnt\_lsb + poc\_lsb\_lt[ i ]  
 else~~ if( used\_by\_curr\_pic\_lt\_flag[ i ] )  
 PocLtCurr[ j++ ] = PocLt[ i ]~~poc\_lsb\_lt[ i ]~~  
 else  
 PocLtFoll[ k++ ] = PocLt[ i ]~~poc\_lsb\_lt[ i ]~~  
NumPocLtCurr = j  
NumPocLtFoll = k

...

# Patent rights declaration(s)

**Qualcomm Incorporated may have current or pending patent rights 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).**