|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11**  10th Meeting: Stockholm, SE, 11 – 20 July 2012 | Document: JCTVC-J0109 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **AHG9: Header parameter set (HPS)** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Ying Chen, Ye-Kui Wang Qualcomm Inc. 5775 Morehouse Drive San Diego, CA 92121 USA  Miska M. Hannuksela Nokia Corporation Visiokatu 3 33720 Tampere Finland | Tel: Email: | 1-858-845-6589 [cheny@qualcomm.com](mailto:cheny@qualcomm.com)  1-858-651-8345 [yekuiw@qualcomm.com](mailto:yekuiw@qualcomm.com)  miska.hannuksela@nokia.com |
| *Source:* | Qualcomm Inc., Nokia Corporation | | |

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

# Abstract

At the previous JCT-VC meeting, the proposal on slice header prediction in JCTVC-I0070 was discussed. It was noted in the meeting notes that using some kind of parameter set to enable slice header prediction seemed promising. Following such a direction, this document proposes a slice header prediction mechanism based on a so-called header parameter set (HPS), with two slightly different alternative approaches. In the first approach (single-AU HPS), an HPS is required to be used within one access unit. In the second approach (multi-AU HPS), an HPS may be used by multiple access units.

It is asserted that the proposed mechanism avoids the drawbacks in the JCTVC-I0070 design, the single-AU HPS approach is expected to provide similar coding gains as reported in JCTVC-I0070, and the multi-AU HPS approach provides opportunities for coding gains with increased complexity.

It is proposed to adopt one of the two approaches of the HPS based slice header prediction mechanism.

# Introduction

A slice header prediction method was proposed in JCTVC-I0070, providing about 1.5% BD-rate reduction on average in low-delay B main configuration compared to HM6.0 when a slice size of about 36 LCUs of size 64x64 was used in both.

The method in JCTVC-I0070 involves carriage of slice headers in the access unit delimiter (AUD) NAL unit and addition of a flag to identify whether a slice header is the same as in the access unit delimiter (AUD) or not. It was commented at the previous JCT-VC meeting that the AUD can only be in one place and cannot be repeated within one access unit. Thus, some kind of parameter set should be used to carry the slice header parameters to be inherited by slice headers. It was noted in the meeting minutes that this parameter set suggestion seemed promising.

In the method in JCTVC-I0070, a slice header may also be predicted from the previous slice in scan order. However, when a pre-encoded bitstream needs to be transported in an environment that has a different amount of losses to the bitstream than the target transmission environment during encoding, one of the following undesirable situations would occur:

* The sender decodes and re-encodes the slice headers to include sufficient repetition of slice header parameters to adapt to the new transmission environment.
* In case a higher amount of error resilience is needed, the sender repeats some entire slices to ensure the reception of the slice header parameters that are needed by other slices, resulting in significantly more overhead.
* The sender does not adapt to transmission environment, so that either slice header parameters are repeated more than necessary (in case a lower amount of error resilience is needed) or less than necessary (in case a higher amount of error resilience is needed).

To address the above drawbacks of the method in JCTVC-I0070, in this document, we propose a slice header prediction mechanism based on a so-called header parameter set (HPS), with two alternative approaches, as described below.

# Proposal

A new parameter set, namely Header Parameter Set (HPS) is introduced. A slice header is predicted from one or more HPSs. If a picture consists of only one slice, the use of HPS is optional and a slice header can be included in the coded slice NAL unit instead. Two alternative approaches of the HPS design are provided in this document:

1. Single-AU HPS: HPS is applicable only to the slices within the same assess unit.
2. Multi-AU HPS: HPS may be applicable to slices in multiple access units.

The two approaches are very similar in their syntax – basically the only syntax difference is that in the multi-AU HPS design, the pic\_order\_cnt and rap\_pic\_id are not present in the HPS syntax. The main differences between the two approaches arise from the fact that the single-AU HPS design requires transmission of an HPS for each access unit, while the multi-AU HPS design allows re-use of the same HPS across multiple AUs. The main differences can be summarized as follows:

1. Compression performance:

It is asserted that for periodic GOP structures only few multi-AU HPSes would be required for the entire coded view sequence, whereas a single-AU HPS is required for each access unit.

1. Detection of HPS NAL unit losses:

The single-AU HPS design enables decoders to detect losses of HPS NAL units as follows:

When an access unit has been decoded, the HPS(es) it referred to are removed or marked unavailable for decoding. If any coded slice refers to an hps\_id value for which an HPS is not available for decoding, the decoder can conclude a loss an HPS NAL unit.

The multi-AU HPS design may require e.g. additional constraints on the use of hps\_id (see e.g. JCTVC-J0072) to detect HPS NAL unit losses in all cases.

It is asserted that the compression benefit of the single-AU HPS approach is similar to that of carrying slice headers in the access unit delimiter NAL unit, as proposed in JCTVC-I0070. The multi-AU HPS approach provides opportunities for more compression benefit.

We propose adoption of one of the two approaches of the HPS based slice header prediction mechanism in the HEVC standard.

## HPS applicable to one AU

### Header parameter set RBSP syntax

|  |  |
| --- | --- |
| header\_parameter\_set( ) { | Descriptor |
| **header\_para\_set\_id** | ue(v) |
| **slice\_type** | ue(v) |
| **common\_info\_present\_flag** | u(1) |
| **//**info. that is common for all slices |  |
| if (common\_info\_present\_flag) |  |
| common\_info\_table( 1 ) |  |
| **//**info. related to reference picture lists |  |
| **ref\_pic\_list\_related\_info\_present\_flag** | u(1) |
| if ( ref\_pic\_list\_related\_info\_present\_flag ) |  |
| reference\_pic\_related\_info\_table( ) |  |
| // prediction weights |  |
| if( ( weighted\_pred\_flag && slice\_type = = P) | |  ( weighted\_bipred\_idc = = 1 && slice\_type = = B ) ) |  |
| { |  |
| **pred\_weight\_table\_present\_flag** | u(1) |
| if (pred\_weight\_table\_present\_flag ) |  |
| pred\_weight\_table( ) |  |
| } |  |
| //deblocking |  |
| if ( deblocking\_filter\_control\_present\_flag ) { |  |
| **deblocking\_para\_table\_present\_flag** | u(1) |
| if (deblocking\_para\_table\_present\_flag) |  |
| deblocking\_para\_table( ) |  |
| } |  |
| // other info. |  |
| if( cabac\_init\_present\_flag && slice\_type != I ) |  |
| **cabac\_init\_flag** | u(1) |
| **slice\_qp\_delta** | se(v) |
| 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(adaptive\_loop\_filter\_enabled\_flag ) |  |
| **aps\_id** | ue(v) |
| if( separate\_colour\_plane\_flag = = 1 ) |  |
| **colour\_plane\_id** | u(2) |
| // extension … |  |
| **hps\_extension\_flag** | u(1) |
| if( hps\_extension\_flag ) |  |
| while( more\_rbsp\_data( ) ) |  |
| **hps\_extension\_data\_flag** | u(1) |
| } |  |

#### Common information table syntax

|  |  |
| --- | --- |
| common\_info\_table( inHpsFlag ) { | Descriptor |
| if( inHpsFlag ){ |  |
| **rap\_pic\_flag** | u(1) |
| if( rap\_pic\_flag ) |  |
| **idr\_pic\_flag** | u(1) |
| } |  |
| **pic\_parameter\_set\_id** | ue(v) |
| if( !hpsIdrPicFlag ) { |  |
| **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( output\_flag\_present\_flag ) |  |
| **pic\_output\_flag** | u(1) |
| if(hpsRapPicFlag) { |  |
| **rap\_pic\_id** | ue(v) |
| **no\_output\_of\_prior\_pics\_flag** | u(1) |
| } |  |
| } |  |

#### Reference pictures related information table syntax

|  |  |
| --- | --- |
| reference\_pic\_related\_info\_table ( ){ | **Descriptor** |
| 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( 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( slice\_type = = P | | slice\_type = = B ) |  |
| **five\_minus\_max\_num\_merge\_cand** | ue(v) |
| } |  |

#### Deblocking parameters table syntax

|  |  |
| --- | --- |
| deblocking\_para\_table( ){ | Descriptor |
| 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) |
| } |  |
| } |  |
| } |  |

### Slice header syntax

|  |  |
| --- | --- |
| slice\_header( ) { | Descriptor |
| **slice\_address** | ue(v) |
| if( slice\_address != 0 ) |  |
| **dependent\_slice\_flag** | u(1) |
| else |  |
| **single\_slice\_no\_hps\_flag** | u(1) |
| if( !single\_slice\_no\_hps\_flag && !dependent\_slice\_flag ) |  |
| **other\_info\_override\_flag** |  |
| if( ( single\_slice\_no\_hps\_flag | | other\_info\_override\_flag )  && !dependent\_slice\_flag ) |  |
| **slice\_type** | ue(v) |
| if( single\_slice\_no\_hps\_flag && !dependent\_slice\_flag ) { |  |
| common\_info\_table( 0 ) |  |
| reference\_pic\_related\_info\_table( ) |  |
| if( ( weighted\_pred\_flag && slice\_type = = P) | |  ( weighted\_bipred\_idc = = 1 && slice\_type = = B ) ) |  |
| pred\_weight\_table( ) |  |
| if ( deblocking\_filter\_control\_present\_flag ) |  |
| deblocking\_para\_table( ) |  |
| } |  |
| if( ( single\_slice\_no\_hps\_flag | | other\_info\_override\_flag )  && !dependent\_slice\_flag ) { |  |
| if( cabac\_init\_present\_flag && slice\_type != I ) |  |
| **cabac\_init\_flag** | u(1) |
| **slice\_qp\_delta** | se(v) |
| 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(adaptive\_loop\_filter\_enabled\_flag ) |  |
| **aps\_id** | ue(v) |
| if( separate\_colour\_plane\_flag = = 1 ) |  |
| **colour\_plane\_id** | u(2) |
| } |  |
| if( !single\_slice\_no\_hps\_flag && !dependent\_slice\_flag ) { |  |
| **common\_info\_hps\_id** | ue(v) |
| **prediction\_from\_one\_hps\_flag** | u(1) |
| if( !prediction\_from\_one\_hps\_flag ) { |  |
| **reference\_pic\_related\_info\_hps\_id** | ue(v) |
| if( ( weighted\_pred\_flag && slice\_type = = P) | |  ( weighted\_bipred\_idc = = 1 && slice\_type = = B ) ) |  |
| **pred\_weight\_table\_hps\_id** | ue(v) |
| if( deblocking\_filter\_control\_present\_flag ) |  |
| **deblocking\_para\_table\_hps\_id** | ue(v) |
| } |  |
| if( ( single\_slice\_no\_hps\_flag | | other\_info\_override\_flag )  && !dependent\_slice\_flag &&  seq\_loop\_filter\_across\_slices\_enabled\_flag &&  ( slice\_adaptive\_loop\_filter\_flag | | slice\_sample\_adaptive\_offset\_flag[ 0 ]  | | !slice\_heder\_disable\_deblocking\_filter\_flag )  ) |  |
| **slice\_loop\_filter\_across\_slices\_enabled\_flag** | u(1) |
| 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( 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( 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) |
| } |  |
| } |  |

### Header parameter set RBSP semantics

The semantics of a syntax element in the HPS, if currently present in slice header in the latest HEVC draft specification, is the same as specified in the latest HEVC draft specification.

**rap\_pic\_flag** specifies the value of the variable hpsRapPicFlag. The value of rap\_pic\_flag shall be equal to RapPicFlag of the coded slice NAL unit referring to the header parameter set that contains the common\_info\_table( ) syntax structure.

The value of hpsRapPicFlag is derived as follows. If rap\_pic\_flag is present, the value of hpsRapPicFlag is set to be equal to rap\_pic\_flag. Otherwise (rap\_pic\_flag is not present), the value of rap\_pic\_flag is set to be equal to RapPicFlag of the coded slice NAL unit for which the slice header contains the common\_info\_table( ) syntax structure.

**idr\_pic\_flag** specifies the value of the variable hpsIdrPicFlag. The value of idr\_pic\_flag shall be equal to IdrPicFlag of the coded slice NAL unit referring to the header parameter set that contains the common\_info\_table( ) syntax structure. When rap\_pic\_flag is present and the value is equal to 0, the value of idr\_pic\_flag is inferred to be equal to 0.

The value of hpsIdrPicFlag is derived as follows. If rap\_pic\_flag is present, the value of hpsIdrPicFlag is set to be equal to the value of idr\_pic\_flag. Otherwise (rap\_pic\_flag is not present), the value of idr\_pic\_flag is set to be equal to IdrPicFlag of the coded slice NAL unit for which the slice header contains the common\_info\_table( ) syntax structure.

**header\_para\_set\_id** identifies a header parameter set.

**common\_info\_present\_flag** equal to 1 specifies that the common\_info\_table( ) syntax structure is present in the current header parameter set. common\_info\_present\_flagequal to 0 specifies that the common\_info\_table( ) syntax structure is not present in the current header parameter set.

**ref\_pic\_list\_related\_info\_present\_flag** equal to 1 specifies that the reference\_pic\_related\_info\_table( ) syntax structure is present in the current header parameter set. ref\_pic\_list\_related\_info\_present\_flag equal to 0 specifies that the reference\_pic\_related\_info\_table( ) syntax structure is not present in the current header parameter set.

**pred\_weight\_table\_present\_flag** equal to 1 specifies that the pred\_weight\_table( ) syntax structure is present in the current header parameter set, pred\_weight\_table\_present\_flag equal to 0 specifies that the pred\_weight\_table( ) syntax structure is not present in the current header parameter set. When not present, this flag is inferred to be equal to 0.

**deblocking\_para\_table\_present\_flag** equal to 1 specifies that the deblocking\_para\_table( ) syntax structure is present in the current header parameter set, deblocking\_para\_table\_present\_flag equal to 0 specifies that the deblocking\_para\_table( ) syntax structure is not present in the current header parameter set. When not present, this flag is inferred to be equal to 0.

**hps\_extension\_flag** equal to 0 specifies that no hps\_extension\_data\_flag syntax elements are present in the RBSP syntax structure. Decoders shall ignore all data that follow the value 1 for hps\_extension\_flag in a NAL unit.

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

### Slice header semantics

**single\_slice\_no\_hps\_flag** equal to 1 specifies that the current picture consists of only one slice and all slice header parameters are directly included in the slice header. single\_slice\_no\_hps\_flagequal to 0 specifies that the current picture may consist of multiple slices and some slice header parameters may be predicted from header parameter sets.

**prediction\_from\_one\_hps\_flag** equal to 1 specifies that the current slice header is predicted from only one HPS. prediction\_from\_one\_hps\_flagequal to 0 specifies that the current slice header may be predicted from multiple HPSs.

**common\_info\_hps\_id** identifies the HPS used to predict the syntax elements in the common\_info\_table( ) syntax structure for the current slice header.

**reference\_pic\_related\_info\_hps\_id** identifies the HPS used to predict the syntax elements in the reference\_pic\_related\_info\_table( ) syntax structure for the current slice header. When not present, this syntax element is inferred to be equal to common\_info\_hps\_id.

**pred\_weight\_table\_hps\_id** identifies the HPS used to predict the syntax elements in the pred\_weight\_table( ) syntax structure for the current slice header. When not present, this syntax element is inferred to be equal to common\_info\_hps\_id.

**deblocking\_para\_table\_hps\_id** identifies the HPS used to predict the syntax elements in the deblocking\_para\_table( ) syntax structure for the current slice header. When not present, this syntax element is inferred to be equal to common\_info\_hps\_id.

**other\_info\_override\_flag** equal to 1 indicates that other syntax elements, including, cabac\_init\_flag, slice\_qp\_delta, slice\_loop\_filter\_across\_slices\_enabled\_flag, and aps\_id in slice header are signalled in the slice header and override the syntax elements in the HPS.

## HPS applicable to multiple AUs

When implementing this solution, only the layout of syntax elements is changed, thus the semantics of the syntax elements are not provided in this section.

More specifically, syntax elements pic\_order\_cnt\_lsb and rap\_pic\_id are moved from common information table to slice header.

### Header parameter set RBSP syntax

No change.

#### Common information table syntax

Changes on top of 2.1.1.1 are marked.

|  |  |
| --- | --- |
| common\_info\_table( inHpsFlag ) { | Descriptor |
| if( inHpsFlag ){ |  |
| **rap\_pic\_flag** | u(1) |
| if( rap\_pic\_flag ) |  |
| **idr\_pic\_flag** | u(1) |
| } |  |
| **pic\_parameter\_set\_id** | ue(v) |
| if( !hpsIdrPicFlag ) { |  |
| **~~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( output\_flag\_present\_flag ) |  |
| **pic\_output\_flag** | u(1) |
| if(hpsRapPicFlag) ~~{~~ |  |
| **~~rap\_pic\_id~~** | ~~ue(v)~~ |
| **no\_output\_of\_prior\_pics\_flag** | u(1) |
| ~~}~~ |  |
| } |  |

#### Reference pictures related information table syntax

No change.

#### Deblocking parameters table syntax

No change.

### Slice header syntax

Changes on top of 2.1.2 are marked.

|  |  |
| --- | --- |
| slice\_header( ) { | Descriptor |
| **slice\_address** | ue(v) |
| if( slice\_address != 0 ) |  |
| **dependent\_slice\_flag** | u(1) |
| else |  |
| **single\_slice\_no\_hps\_flag** | u(1) |
| if( !single\_slice\_no\_hps\_flag && !dependent\_slice\_flag ) |  |
| **other\_info\_override\_flag** |  |
| if( ( single\_slice\_no\_hps\_flag | | other\_info\_override\_flag )  && !dependent\_slice\_flag ) |  |
| **slice\_type** | ue(v) |
| if( ~~single\_slice\_no\_hps\_flag &&~~ !dependent\_slice\_flag ){ |  |
| if( single\_slice\_no\_hps\_flag ) |  |
| common\_info\_table(0 ) |  |
| else |  |
| **common\_info\_hps\_id** | ue(v) |
| } |  |
| **pic\_order\_cnt\_lsb** | u(v) |
| if( RapPicFlag ) |  |
| **rap\_pic\_id** | ue(v) |
| if( single\_slice\_no\_hps\_flag && !dependent\_slice\_flag ){ |  |
| reference\_pic\_related\_info\_table ( ) |  |
| if( ( weighted\_pred\_flag && slice\_type = = P) | |  ( weighted\_bipred\_idc = = 1 && slice\_type = = B ) ) |  |
| pred\_weight\_table( ) |  |
| if ( deblocking\_filter\_control\_present\_flag ) |  |
| deblocking\_para\_table( ) |  |
| } |  |
| if( ( single\_slice\_no\_hps\_flag | | other\_info\_override\_flag )  && !dependent\_slice\_flag ) { |  |
| if( cabac\_init\_present\_flag && slice\_type != I ) |  |
| **cabac\_init\_flag** | u(1) |
| **slice\_qp\_delta** | se(v) |
| 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(adaptive\_loop\_filter\_enabled\_flag ) |  |
| **aps\_id** | ue(v) |
| if( separate\_colour\_plane\_flag = = 1 ) |  |
| **colour\_plane\_id** | u(2) |
| } |  |
| if( !single\_slice\_no\_hps\_flag && !dependent\_slice\_flag ) { |  |
| **~~common\_info\_hps\_id~~** | ~~ue(v)~~ |
| **prediction\_from\_one\_hps\_flag** | u(1) |
| if( !prediction\_from\_one\_hps\_flag ) { |  |
| **reference\_pic\_related\_info\_hps\_id** | ue(v) |
| if( ( weighted\_pred\_flag && slice\_type = = P) | |  ( weighted\_bipred\_idc = = 1 && slice\_type = = B ) ) |  |
| **pred\_weight\_table\_hps\_id** | ue(v) |
| if( deblocking\_filter\_control\_present\_flag ) |  |
| **deblocking\_para\_table\_hps\_id** | ue(v) |
| } |  |
| if( ( single\_slice\_no\_hps\_flag | | other\_info\_override\_flag )  && !dependent\_slice\_flag &&  seq\_loop\_filter\_across\_slices\_enabled\_flag &&  ( slice\_adaptive\_loop\_filter\_flag | | slice\_sample\_adaptive\_offset\_flag[ 0 ]  | | !slice\_heder\_disable\_deblocking\_filter\_flag )  ) |  |
| **slice\_loop\_filter\_across\_slices\_enabled\_flag** | u(1) |
| 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( 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( 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) |
| } |  |
| } |  |

# 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).**

**Nokia Corporation 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).**