|  |  |
| --- | --- |
| **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-J0124 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **AHG10: Video parameter set for HEVC extensions** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Information | | |
| *Author(s) or Contact(s):* | Ying Chen Ye-Kui Wang  5775 Morehouse Drive San Diego, CA 92121 USA | Tel: Email: | 1-858-845-6589 [cheny@qualcomm.com](mailto:cheny@qualcomm.com)  1-858-651-8345 [yekuiw@qualcomm.com](mailto:yekuiw@qualcomm.com) |
| *Source:* | Qualcomm Incorporated | | |

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

# Abstract

Video parameter set (VPS) was adopted into HEVC, and includes mainly sequence-level temporal scalability related information. In JCTVC-J0114, VPS syntax design with corresponding changes in SPS (including VUI), is proposed to enable the use of VPS in session negotiation as well as to reduce the number of bits needed for the representation of SPSs. To enable more convenient session negotiation for various operation points in the HEVC extensions and better bit efficiency for the coding of parameter sets, an example design of the VPS for future extensions based on the VPS proposed in JCTVC-J0114 is included in this document to show the extensibility of the design as shown in JCTVC-J0114.

# Introduction

Currently video parameter set (VPS) contains mainly sequence-level temporal scalability related information, with the following syntax:

|  |  |
| --- | --- |
| video\_parameter\_set\_rbsp( ) { | Descriptor |
| **vps\_max\_temporal\_layers\_minus1** | u(3) |
| **vps\_max\_layers\_minus1** | u(5) |
| **video\_parameter\_set\_id** | ue(v) |
| **vps\_temporal\_id\_nesting\_flag** | u(1) |
| for( i = 0; i <= vps\_max\_temporal\_layers\_minus1; i++ ) { |  |
| **vps\_max\_dec\_pic\_buffering[** i **]** | ue(v) |
| **vps\_num\_reorder\_pics[** i **]** | ue(v) |
| **vps\_max\_latency\_increase[** i **]** | ue(v) |
| } |  |
| **vps\_extension\_flag** | u(1) |
| if( vps\_extension\_flag ) |  |
| while( more\_rbsp\_data( ) ) |  |
| **vps\_extension\_data\_flag** | u(1) |
| } |  |
| rbsp\_trailing\_bits( ) |  |
| } |  |

A significant amount of the information contained in SPSs (including VUI) is typically the same for all the layers or views. When all the information is exactly the same, then per the current design, two copies would be needed, for the base layer/view, one for enhancement layers/views (as they can share the same SPS then). If any part of the SPS information changes from one enhancement layer to another enhancement layer, then the rest of the SPS information needs also to repeated once more. SPSs in many application scenarios are transmitted out-of-band, which means the smaller the overall size of all the SPSs the shorter the initial delay, as out-of-band transmission is reliable at the cost of increased initial delay in error-prone environment, wherein whenever there is a loss then retransmission would have to be applied. If signalled in-band in broadcast or multicast applications, repetitions of SPSs would be needed at RAP access units to enable tune-in.

In MVC and SVC, most of the syntax elements in SPSs of different layers are identical, so making them present in VPS can significantly reduce the number of bits needed altogether for VPS and SPS.

A challenge in both MVC and SVC design is the indication of operation points, each of which requires the definition of the operation point (e.g., in MVC, a list of target output views in a certain temporal level) itself as well as the characteristics.

In JCTVC-J0114, we proposed a modified design to avoid repetition of SPS information. The proposed design includes all information that is essential for session negotiation to be easily accessible in two ways. Firstly, such information is gathered together and put as early as possible in the VPS. Secondly, such information can be accessed without the need of entropy decoding.

In this contribution, we further extend the VPS design in HEVC base specification to HEVC extensions. The VPS extension is designed in a way that retrieving the information that is critical to session negotiation for various operation points doesn’t require entropy decoding. Furthermore, in case a set of specific groups of information signalled in the VPS base syntax is not sufficient, additional sets are signalled in the VPS extension.

The proposed syntax and semantics are described below.

# Proposal

## Syntax

Note that only the video parameter set RBSP syntax is different than that in JVCVC-J0114, due to the extension of the VPS. All the other syntax tables do not have any changes compared to JVCVC-J0114.

### Video parameter set RBSP syntax

|  |  |
| --- | --- |
| video\_parameter\_set\_rbsp( ) { | Descriptor |
| **vps\_max\_temporal\_layers\_minus1** | u(3) |
| **vps\_max\_layers\_minus1** | u(5) |
| profile\_level\_info( 0, vps\_max\_temporal\_layers\_minus1 ) |  |
| **video\_parameter\_set\_id** | u(5) |
| **vps\_temporal\_id\_nesting\_flag** | u(1) |
| rep\_format\_info( 0, 0 ) |  |
| bitrate\_framerate\_info( 0, vps\_max\_temporal\_layers\_minus1 ) |  |
| **next\_essential\_info\_byte\_offset** | u(12) |
| rep\_format\_info( 0, 1 ) |  |
| for( i = 0; i <= vps\_max\_temporal\_layers\_minus1; i++ ) { |  |
| **vps\_max\_dec\_pic\_buffering[** i **]** | ue(v) |
| **vps\_num\_reorder\_pics[** i **]** | ue(v) |
| **vps\_max\_latency\_increase[** i **]** | ue(v) |
| } |  |
| hrd\_info( 0, vps\_max\_temporal\_layers\_minus1 ) |  |
| vui\_vps\_set( 0 ) |  |
| **num\_vps\_short\_term\_ref\_pic\_sets** | ue(v) |
| for( i = 0; i < num\_vps\_short\_term\_ref\_pic\_sets; i++ ) |  |
| short\_term\_ref\_pic\_set( i ) |  |
| optional\_sps\_parameters( ) |  |
| **~~vps\_extension\_flag~~** | ~~u(1)~~ |
| **bit\_equal\_to\_one** | u(1) |
| vps\_extension( ) |  |
| **vps\_extension\_flag** | u(1) |
| if( vps\_extension\_flag ) |  |
| while( more\_rbsp\_data( ) ) |  |
| **vps\_extension\_data\_flag** | u(1) |
| rbsp\_trailing\_bits( ) |  |
| } |  |

#### Profile and level information table syntax

|  |  |
| --- | --- |
| profile\_level\_info( index, NumTempLevelMinus1 ) { |  |
| **profile\_space** | u(3) |
| **profile\_idc** | u(5) |
| for( j = 0; j < 32; j++ ) |  |
| **profile\_compatability\_flag[** I **]** | u(1) |
| **constraint\_flags** | u(16) |
| **level\_idc** | u(8) |
| **level\_lower\_temporal\_layers\_present\_flag** | u(1) |
| if( level\_lower\_temporal\_layers\_present\_flag ) |  |
| for ( i = 0; i < NumTempLevelMinus1; i++ ) |  |
| **level\_idc**[ i ] | u(8) |
| profileLevelInfoIdx = index |  |
| } |  |

#### Representation format information table syntax

|  |  |
| --- | --- |
| rep\_format\_info( index, partIdx ) { |  |
| if( !partIdx ){ |  |
| **chroma\_format\_idc** | u(2) |
| if( chroma\_format\_idc = = 3 ) |  |
| **separate\_colour\_plane\_flag** | u(1) |
| **bit\_depth\_luma\_minus8** | u(2) |
| **bit\_depth\_chroma\_minus8** | u(2) |
| **pic\_width\_in\_luma\_samples** | u(16) |
| **pic\_height\_in\_luma\_samples** | u(16) |
| } |  |
| else { |  |
| **pic\_cropping\_flag** | u(1) |
| if( pic\_cropping\_flag ) { |  |
| **pic\_crop\_left\_offset** | ue(v) |
| **pic\_crop\_right\_offset** | ue(v) |
| **pic\_crop\_top\_offset** | ue(v) |
| **pic\_crop\_bottom\_offset** | ue(v) |
| } |  |
| **}** |  |
| repFormatInfoIdx = index |  |
| } |  |

#### Bitrate and frame rate information table syntax

|  |  |
| --- | --- |
| bitrate\_framerate\_info( TempLevelLow, TempLevelHigh ){ |  |
| **for( i = TempLevelLow; i <= TempLevelHigh; i++ ) {** |  |
| **bitrate\_info\_present\_flag[ i ]** | u(1) |
| **frm\_rate\_info\_present\_flag[ i ]** | u(1) |
| **if( bitrate\_info\_present\_flag[ i ] ) {** |  |
| **avg\_bitrate**[ i ] | u(16) |
| **max\_bitrate** [ i ] | u(16) |
| } |  |
| if( frm\_rate\_info\_present\_flag[ i ] ) { |  |
| **constant\_frm\_rate\_idc**[ i ] | u(2) |
| **avg\_frm\_rate**[ i ] | u(16) |
| } |  |
| } |  |
| } |  |

#### HRD information table syntax

|  |  |
| --- | --- |
| hrd\_info( TempLevelLow, TempLevelHigh ) { |  |
| for ( i = TempLevelLow, nalHrdPresent = 0, vclHrdPresent = 0;  i <= NumTempLevelMinus1; i++ ) { |  |
| **nal\_hrd\_parameters\_present\_flag[** i **]** | u(1) |
| if( nal\_hrd\_parameters\_present\_flag[ i ] ) { |  |
| hrd\_parameters( nalHrdPresent ) |  |
| nalHrdPresent++ |  |
| } |  |
| **vcl\_hrd\_parameters\_present\_flag[** i **]** | u(1) |
| if( vcl\_hrd\_parameters\_present\_flag[ i ] ) { |  |
| hrd\_parameters( vclHrdPresent ) |  |
| vclHrdPresent++ |  |
| } |  |
| if( nalHrdPresent + vclHrdPresent = = 1 ) { |  |
| **low\_delay\_hrd\_flag** | u(1) |
| **sub\_pic\_cpb\_params\_present\_flag** | u(1) |
| **num\_units\_in\_sub\_tick** | u(32) |
| } |  |
|  |  |
| } |  |

#### VUI VPS set table syntax

|  |  |
| --- | --- |
| vui\_vps\_set( index ) { |  |
| **vui\_video\_parameters\_present\_flag** | u(1) |
| if( vui\_video\_parameters\_present\_flag ) |  |
| vui\_parameters() |  |
| vuiVpsSetIndex = index |  |
| } |  |

#### Optional SPS parameters syntax

|  |  |
| --- | --- |
| optional\_sps\_parameters( ) { |  |
| **pcm\_enabled\_flag** | u(1) |
| if( pcm\_enabled\_flag ) { |  |
| **pcm\_sample\_bit\_depth\_luma\_minus1** | u(4) |
| **pcm\_sample\_bit\_depth\_chroma\_minus1** | u(4) |
| } |  |
| **log2\_max\_pic\_order\_cnt\_lsb\_minus4** | ue(v) |
| ~~for( i = 0; i <= sps\_max\_temporal\_layers\_minus1; i++ ) {~~ |  |
| **~~sps\_max\_dec\_pic\_buffering[~~**~~i~~**~~]~~** | ~~ue(v)~~ |
| **~~sps\_num\_reorder\_pics[~~**~~i~~**~~]~~** | ~~ue(v)~~ |
| **~~sps\_max\_latency\_increase[~~**~~i~~**~~]~~** | ~~ue(v)~~ |
| ~~}~~ |  |
| **restricted\_ref\_pic\_lists\_flag** | u(1) |
| if( restricted\_ref\_pic\_lists\_flag ) |  |
| **lists\_modification\_present\_flag** | u(1) |
| **log2\_min\_coding\_block\_size\_minus3** | ue(v) |
| **log2\_diff\_max\_min\_coding\_block\_size** | ue(v) |
| **log2\_min\_transform\_block\_size\_minus2** | ue(v) |
| **log2\_diff\_max\_min\_transform\_block\_size** | ue(v) |
| if( pcm\_enabled\_flag ) { |  |
| **log2\_min\_pcm\_coding\_block\_size\_minus3** | ue(v) |
| **log2\_diff\_max\_min\_pcm\_coding\_block\_size** | ue(v) |
| } |  |
| **max\_transform\_hierarchy\_depth\_inter** | ue(v) |
| **max\_transform\_hierarchy\_depth\_intra** | ue(v) |
| **scaling\_list\_enable\_flag** | u(1) |
| if( scaling\_list\_enable\_flag ) { |  |
| **sps\_scaling\_list\_data\_present\_flag** | u(1) |
| if( sps\_scaling\_list\_data\_present\_flag ) |  |
| scaling\_list\_param( ) |  |
| } |  |
| **chroma\_pred\_from\_luma\_enabled\_flag** | u(1) |
| **transform\_skip\_enabled\_flag** | u(1) |
| **seq\_loop\_filter\_across\_slices\_enabled\_flag** | u(1) |
| **asymmetric\_motion\_partitions\_enabled\_flag** | u(1) |
| **nsrqt\_enabled\_flag** | u(1) |
| **sample\_adaptive\_offset\_enabled\_flag** | u(1) |
| **adaptive\_loop\_filter\_enabled\_flag** | u(1) |
| if( adaptive\_loop\_filter\_enabled\_flag ) |  |
| **alf\_coef\_in\_slice\_flag** | u(1) |
| if( pcm\_enabled\_flag ) |  |
| **pcm\_loop\_filter\_disable\_flag** | u(1) |
| **~~sps\_temporal\_id\_nesting\_flag~~** | ~~u(1)~~ |
| [Ed. (BB): x y padding syntax missing here, present in HM software ] |  |
| if( log2\_min\_coding\_block\_size\_minus3 = = 0 ) |  |
| **inter\_4x4\_enabled\_flag** | u(1) |
| **long\_term\_ref\_pics\_present\_flag** | u(1) |
| **sps\_temporal\_mvp\_enable\_flag** | u(1) |
| **tiles\_fixed\_structure\_idc** | u(2) |
| } |  |

#### VPS extension syntax

|  |  |
| --- | --- |
| vps\_extension( ) { |  |
| **byte\_alligned\_bits** | u(v) |
| **num\_additional\_profile\_level\_info** | u(4) |
| **num\_additional\_rep\_fromat\_info** | u(3) |
| **num\_additional\_dependency\_operation\_points** | u(8) |
| **extension\_type** | u(3) |
| for( i =0; i< num\_additional\_profile\_level\_info; i++ ) |  |
| profile\_level\_info( i + 1, vps\_max\_temporal\_layers\_minus1 ) |  |
| for( i = 0; i < num\_additional\_rep\_fromat\_info; i++ ) |  |
| rep\_format\_info( i+1, 0 ) |  |
| for (k=0; k< num\_additional\_dependency\_operation\_points ;k++) { |  |
| if( num\_additional\_profile\_level\_info ) |  |
| **profile\_level\_index[** k **]** | u(4) |
| if( num\_additional\_rep\_fromat\_info ) |  |
| **ref\_format\_index[** k **]** | u(3) |
| **applicable\_lowest\_temporal\_id[** k **]** | u(3) |
| **applicable\_highest\_temporal\_id[** k **]** | u(3) |
| } |  |
| for (k=0; k< num\_additional\_dependency\_operation\_points; k++) { |  |
| bitrate\_framerate\_info( applicable\_lowest\_temporal\_id[ k ],   applicable\_highest\_temporal\_id[ k ] ) |  |
| } |  |
| // layer dependency |  |
| for (k=0; k< num\_additional\_dependency\_operation\_points; k++) { |  |
| if( extension\_type = = 0 ) { /\* Condition always true for 3DV \*/ |  |
| **depth\_included\_flag[** k **]** | u(1) |
| **num\_target\_output\_views\_minus1[** k **]** | u(5) |
| **num\_depedent\_layers[** k **]** | u(5) |
| for( j = 0; j < num\_target\_output\_views\_minus1[ k ]; j++ ) |  |
| **layer\_id[** k **][** j **]** | u(5) |
| for( j = 0; j < num\_depedent\_layers[ k ]; j++ ) |  |
| **dependent\_layer\_id[** k **][** j **]** | u(5) |
| } |  |
| else if( extension\_type = = 1) |  |
| **layer\_id[** k **]** | u(5) |
| } |  |
| for( i = 0; i < num\_additional\_rep\_fromat\_info; i++ ) { |  |
| rep\_format\_info( i + 1, 1 ) |  |
| // boundary of the fixed-length and ue(v) |  |
| //vui |  |
| **num\_additional\_vui\_vps\_set\_info** | ue(v) |
| for( i = 0; i < num\_additional\_vui\_vps\_set\_info; i++ ) |  |
| vui\_vps\_set( i + 1 ) |  |
| for (k=0; k< num\_additional\_dependency\_operation\_points; k++) { |  |
| if (num\_additional\_vui\_vps\_set\_info) |  |
| **vui\_vps\_set\_idx** | ue(v) |
| hrd\_info(applicable\_lowest\_temporal\_id[ k ],   applicable\_highest\_temporal\_id[ k ]) |  |
| } |  |
| } |  |

### Sequence parameter set RBSP syntax

|  |  |
| --- | --- |
| seq\_parameter\_set\_rbsp( ) { | Descriptor |
| **~~profile\_space~~** | ~~u(3)~~ |
| **~~profile\_idc~~** | ~~u(5)~~ |
| **~~constraint\_flags~~** | ~~u(16)~~ |
| **~~level\_idc~~** | ~~u(8)~~ |
| ~~for( i = 0; i < 32; i++ )~~ |  |
| **~~profile\_compatability\_flag[~~**~~i~~**~~]~~** | ~~u(1)~~ |
| **seq\_parameter\_set\_id** | ue(v) |
| **video\_parameter\_set\_id** | ue(v) |
| **~~chroma\_format\_idc~~** | ~~ue(v)~~ |
| ~~if( chroma\_format\_idc = = 3 )~~ |  |
| **~~separate\_colour\_plane\_flag~~** | ~~u(1)~~ |
| **~~sps\_max\_temporal\_layers\_minus1~~** | ~~u(3)~~ |
| **~~pic\_width\_in\_luma\_samples~~** | ~~ue(v)~~ |
| **~~pic\_height\_in\_luma\_samples~~** | ~~ue(v)~~ |
| **~~pic\_cropping\_flag~~** | ~~u(1)~~ |
| ~~if( pic\_cropping\_flag ) {~~ |  |
| **~~pic\_crop\_left\_offset~~** | ~~ue(v)~~ |
| **~~pic\_crop\_right\_offset~~** | ~~ue(v)~~ |
| **~~pic\_crop\_top\_offset~~** | ~~ue(v)~~ |
| **~~pic\_crop\_bottom\_offset~~** | ~~ue(v)~~ |
| ~~}~~ |  |
| **~~bit\_depth\_luma\_minus8~~** | ~~ue(v)~~ |
| **~~bit\_depth\_chroma\_minus8~~** | ~~ue(v)~~ |
| ~~[Ed. (BB): chroma bit depth present in HM software but not used further ]~~ |  |
| **num\_short\_term\_ref\_pic\_sets** | ue(v) |
| **use\_rps\_from\_vps\_flag** | u(1) |
| for( i = 0; i < num\_short\_term\_ref\_pic\_sets; i++){ |  |
| idx = use\_rps\_from\_vps\_flag ? num\_vps\_short\_term\_ref\_pic\_sets + i : i |  |
| short\_term\_ref\_pic\_set( idx ) |  |
| } |  |
| **sps\_parameters\_override\_flag** | u(1) |
| if( sps\_parameters\_override\_flag ) |  |
| optional\_sps\_parameters( ) |  |
| **~~vui\_parameters\_present\_flag~~** | ~~u(1)~~ |
| ~~if( vui\_parameters\_present\_flag )~~ |  |
| ~~vui\_parameters( )~~ |  |
| **sps\_extension\_flag** | u(1) |
| if( sps\_extension\_flag ) |  |
| while( more\_rbsp\_data( ) ) |  |
| **sps\_extension\_data\_flag** | u(1) |
| rbsp\_trailing\_bits( ) |  |
| } |  |

### VUI parameters syntax

|  |  |
| --- | --- |
| vui\_parameters( ) { | Descriptor |
| **aspect\_ratio\_info\_present\_flag** | u(1) |
| if( aspect\_ratio\_info\_present\_flag ) { |  |
| **aspect\_ratio\_idc** | u(8) |
| if( aspect\_ratio\_idc = = Extended\_SAR ) { |  |
| **sar\_width** | u(16) |
| **sar\_height** | u(16) |
| } |  |
| } |  |
| **overscan\_info\_present\_flag** | u(1) |
| if( overscan\_info\_present\_flag ) |  |
| **overscan\_appropriate\_flag** | u(1) |
| **video\_signal\_type\_present\_flag** | u(1) |
| if( video\_signal\_type\_present\_flag ) { |  |
| **video\_format** | u(3) |
| **video\_full\_range\_flag** | u(1) |
| **colour\_description\_present\_flag** | u(1) |
| if( colour\_description\_present\_flag ) { |  |
| **colour\_primaries** | u(8) |
| **transfer\_characteristics** | u(8) |
| **matrix\_coefficients** | u(8) |
| } |  |
| } |  |
| **chroma\_loc\_info\_present\_flag** | u(1) |
| if( chroma\_loc\_info\_present\_flag ) { |  |
| **chroma\_sample\_loc\_type\_top\_field** | ue(v) |
| **chroma\_sample\_loc\_type\_bottom\_field** | ue(v) |
| } |  |
| **neutral\_chroma\_indication\_flag** | u(1) |
| **field\_seq\_flag** | u(1) |
| **timing\_info\_present\_flag** | u(1) |
| if( timing\_info\_present\_flag ) { |  |
| **num\_units\_in\_tick** | u(32) |
| **time\_scale** | u(32) |
| **fixed\_pic\_rate\_flag** | u(1) |
| } |  |
| **~~nal\_hrd\_parameters\_present\_flag~~** | ~~u(1)~~ |
| ~~if( nal\_hrd\_parameters\_present\_flag )~~ |  |
| ~~hrd\_parameters( )~~ |  |
| **~~vcl\_hrd\_parameters\_present\_flag~~** | ~~u(1)~~ |
| ~~if( vcl\_hrd\_parameters\_present\_flag )~~ |  |
| ~~hrd\_parameters( )~~ |  |
| ~~if( nal\_hrd\_parameters\_present\_flag | | vcl\_hrd\_parameters\_present\_flag )~~ |  |
| **~~low\_delay\_hrd\_flag~~** | ~~u(1)~~ |
| **~~sub\_pic\_cpb\_params\_present\_flag~~** | ~~u(1)~~ |
| ~~if( sub\_pic\_cpb\_params\_present\_flag )~~ |  |
| **~~num\_units\_in\_sub\_tick~~** | ~~u(32)~~ |
| ~~}~~ |  |
| **bitstream\_restriction\_flag** | u(1) |
| if( bitstream\_restriction\_flag ) { |  |
| **~~tiles\_fixed\_structure\_flag~~** | ~~u(1)~~ |
| **motion\_vectors\_over\_pic\_boundaries\_flag** | u(1) |
| **max\_bytes\_per\_pic\_denom** | ue(v) |
| **max\_bits\_per\_mincu\_denom** | ue(v) |
| **log2\_max\_mv\_length\_horizontal** | ue(v) |
| **log2\_max\_mv\_length\_vertical** | ue(v) |
| } |  |
| } |  |

### HRD parameters syntax

|  |  |
| --- | --- |
| hrd\_parameters( i ) { | Descriptor |
| if( i = = 0 ) { |  |
| **cpb\_cnt\_minus1** | ue(v) |
| **bit\_rate\_scale** | u(4) |
| **cpb\_size\_scale** | u(4) |
| } |  |
| for( SchedSelIdx = 0; SchedSelIdx <= cpb\_cnt\_minus1; SchedSelIdx++ ) { |  |
| **bit\_rate\_value\_minus1[** i **][** SchedSelIdx **]** | ue(v) |
| **cpb\_size\_value\_minus1[** i **] [** SchedSelIdx **]** | ue(v) |
| if( i = = 0 ) |  |
| **cbr\_flag[** SchedSelIdx **]** | u(1) |
| } |  |
| if( i = = 0 ) { |  |
| **initial\_cpb\_removal\_delay\_length\_minus1** | u(5) |
| **cpb\_removal\_delay\_length\_minus1** | u(5) |
| **dpb\_output\_delay\_length\_minus1** | u(5) |
| **time\_offset\_length** | u(5) |
| **}** |  |
| } |  |

### Slice header syntax

|  |  |
| --- | --- |
| slice\_header( ) { | Descriptor |
| ... |  |
| **short\_term\_ref\_pic\_set\_sps\_flag** | u(1) |
| if( !short\_term\_ref\_pic\_set\_sps\_flag ) |  |
| short\_term\_ref\_pic\_set( NumShortTermRefPicSets ) |  |
| else |  |
| **short\_term\_ref\_pic\_set\_idx** | u(v) |
| ... |  |
| } |  |

## Semantics

### Video parameter set RBSP semantics

Note that only the semantics for the syntax elements in the extension part is provided.

**byte\_alligned\_bits** specifies the possible bits that make the bits in the VPS NAL unit prior to num\_additional\_profile\_level\_info byte aligned. byte\_alligned\_bits is in the range of 0 to 7, inclusive.

**num\_additional\_profile\_level\_info** specifies the number of additional profile and level information tables present in the VPS.

**num\_additional\_rep\_fromat\_info** specifies the number of additional Representation format information tables present in the VPS.

**num\_additional\_dependency\_operation\_points** specifies the number of dependency operation points further present in the bitstream, regardless of temporal scalability. Each dependency operation point may include temporal sub operation points, each have the same layer structure.

**extension\_type** specifies the type of the extension of the current bitstream, with 0 corresponding to 3DV and 1 corresponding to SVC.

**profile\_level\_index[** k **]** indicates the index to the level information table signalled in the VPS for the current k-th dependency operation point.

**ref\_format\_index** indicates the index to the representation format information table signalled in the VPS for the current k-th dependency operation point.

**applicable\_lowest\_temporal\_id[** k **]** and **applicable\_highest\_temporal\_id[** k **]** specify respectively the lowest temporal\_id value and the highest temporal\_id value corresponding to the signalled temporal sub operation points of the k-th dependency operation point.

**depth\_included\_flag[** k **]**equal to 1 indicates that the current 3DV dependency operation point contains depth. This flag equal to 0 indicates that the current 3DV operation point does not contain depth.

**num\_target\_output\_views\_minus1[** k **]** plus 1 specifies the number of target output views in the k-th dependency operation point.

**num\_dependent\_layers[** k **]** indicates the number of dependent layers for decoding the current k-th dependency operation point.

**layer\_id[** k **][** j **]** indicates the layer identification of the j-th target output view of the k-th dependency operation point.

**dependent\_layer\_id[** k **][** j **]** indicates the layer identification of the j-th dependent view of the k-the dependency operation point.

**layer\_id[** k **]** identifies the highest layer of the current k-th (SVC) dependency operation point.

**num\_additional\_vui\_vps\_set\_info** specifies the number of additional VUI VPS set table present in the VPS.

# 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).**