Chroma QP adjustment spec, based upon N1005\_v3

This document contains an updated specification text, taking into account the meeting notes decision:

*A table of up to 6 values at the picture level, in which the offsets applied for Cb and Cr can differ (two contexts, unary coding the index)*

#### Picture parameter set RBSP syntax

|  |  |
| --- | --- |
| pic\_parameter\_set\_rbsp( ) { | Descriptor |
| **...** | ... |
| **pps\_extension1\_flag** | u(1) |
| if( pps\_extension1\_flag ) { |  |
| if( transform\_skip\_enabled\_flag ) |  |
| **log2\_transform\_skip\_max\_size\_minus2** | ue(v) |
| **chroma\_qp\_adjustment\_enabled\_flag** | u(1) |
| if( chroma\_qp\_adjustment\_enabled\_flag ) { |  |
| **diff\_cu\_chroma\_qp\_adjustment\_depth** | ue(v) |
| **chroma\_qp\_adjustment\_table\_size\_minus1** | ue(v) |
| for( i = 0; i <= chroma\_qp\_adjustment\_table\_size\_minus1; i++ ) | ue(v) |
| **cb\_qp\_adjustment[** i **]** | se(v) |
| **cr\_qr\_adjustment[** i **]** | se(v) |
| } |  |
| **pps\_extension2\_flag** | u(1) |
| } |  |
| ... |  |
| } |  |

#### Picture parameter set RBSP semantics

**chroma\_qp\_adjustment\_enabled\_flag** equal to 1 specifies that the cu\_chroma\_qp\_adjustment\_flag may be present in the transform unit syntax. cu\_chroma\_qp\_adjustment\_enabled\_flag equal to 0 specifies that the cu\_chroma\_qp\_adjustment\_flag is not present in the transform unit syntax.

When ChromaArrayType is equal to 0, it is a requirement of bitstream conformance that the value of cu\_chroma\_qp\_adjustment\_enabled\_flag shall be equal to 0.

**diff\_cu\_chroma\_qp\_adjustment\_depth** specifies the difference between the luma coding tree block size and the minimum luma coding block size of coding units that convey cu\_chroma\_qp\_adjustment\_flag. The value of diff\_cu\_chroma\_qp\_adjustment\_depth shall be in the range of 0 to log2\_diff\_max\_min\_luma\_coding\_block\_size, inclusive.

XXX: When not present, the value of diff\_cu\_chroma\_qp\_adjustment\_depth is inferred to be equal to 0.

The variable Log2MinCuChromaQpAdjustmentSize is derived as follows:

Log2MinCuQpChromaAdjustmentSize = CtbLog2SizeY − diff\_cu\_chroma\_qp\_adjustment\_depth (7‑31)

**chroma\_qp\_adjustment\_table\_size\_minus1** plus 1 specifies the size of the chroma\_qp\_adjustment[ ] table. The value of chroma\_qp\_adjustment\_table\_size\_minus1 shall be in the range of 0 to 6, inclusive.

**cb\_qp\_adjustment**[ i ] and **cr\_qp\_adjustment**[ i ] specify offsets used in the derivation of Qp′Cb and Qp′Cr respectively. The values of **cb\_qp\_adjustment**[ i ] and **cr\_qp\_adjustment**[ i ] shall be in the range of −12 to +12, inclusive. When ChromaArrayType is equal to 0, **cb\_qp\_adjustment**[ i ] and **cr\_qp\_adjustment**[ i ] are not used in the decoding process and decoders shall ignore their value.

### Slice segment header syntax

#### General slice segment header syntax

|  |  |
| --- | --- |
| slice\_segment\_header( ) { | Descriptor |
| ... |  |
| if( !dependent\_slice\_segment\_flag ) { |  |
| ... |  |
| **slice\_qp\_delta** | se(v) |
| if( pps\_slice\_chroma\_qp\_offsets\_present\_flag ) { |  |
| **slice\_cb\_qp\_offset** | se(v) |
| **slice\_cr\_qp\_offset** | se(v) |
| } |  |
| if( chroma\_qp\_adjustment\_enabled\_flag ) { |  |
| **slice\_chroma\_qp\_adjustment\_enabled\_flag** | u(1) |
| } |  |
| ... |  |
| } |  |
| ... |  |
| if( slice\_segment\_header\_extension\_present\_flag ) { |  |
| **slice\_segment\_header\_extension\_length** | ue(v) |
| for( i = 0; i < slice\_segment\_header\_extension\_length; i++) |  |
| **slice\_segment\_header\_extension\_data\_byte**[ i ] | u(8) |
| } |  |
| byte\_alignment( ) |  |
| } |  |

#### General slice segment header semantics

**slice\_chroma\_qp\_adjustment\_enabled\_flag** equal to 1 specifies that the cu\_chroma\_qp\_adjustment\_flag may be present in the transform unit syntax. cu\_chroma\_qp\_adjustment\_enabled\_flag equal to 0 specifies that the cu\_chroma\_qp\_adjustment\_flag is not present in the transform unit syntax.

When not present, the value of slice\_chroma\_qp\_adjustment\_enabled\_flag shall be inferred to be equal to 0.

#### Coding quadtree syntax

|  |  |
| --- | --- |
| coding\_quadtree( x0, y0, log2CbSize, cqtDepth ) { | Descriptor |
| ... | ... |
| if( cu\_qp\_delta\_enabled\_flag && log2CbSize >= Log2MinCuQpDeltaSize ) { |  |
| IsCuQpDeltaCoded = 0 |  |
| CuQpDeltaVal = 0 |  |
| } |  |
| if( slice\_chroma\_qp\_adjustment\_enabled\_flag   && log2CbSize >= Log2MinCuChromaQpAdjustmentSize ) { |  |
| IsCuChromaQpAdjustmentCoded = 0 |  |
| CuChromaQpAdjustment = 0 |  |
| } |  |
| ... |  |
| } |  |

#### Transform unit syntax

|  |  |
| --- | --- |
| transform\_unit( x0, y0, xBase, yBase, log2TrafoSize, trafoDepth, blkIdx ) { | Descriptor |
| log2TrafoSizeC = log2TrafoSize − ( ChromaArrayType = = 3 ? 0 : 1 ) |  |
| cbfLuma = cbf\_luma[ x0 ][ y0 ][ trafoDepth ] |  |
| cbfChroma =   cbf\_cb[ x0 ][ y0 ][ trafoDepth ] | |   cbf\_cr[ x0 ][ y0 ][ trafoDepth ] | |   ( ChromaArrayType = = 2 &&   ( cbf\_cb[ x0 ][ y0 + ( 1 << log2TrafoSizeC ) ][ trafoDepth ] | |   cbf\_cr[ x0 ][ y0 + ( 1 << log2TrafoSizeC ) ][ trafoDepth ] ) ) |  |
| if( cbfLuma | | cbfChroma ) { |  |
| if( cu\_qp\_delta\_enabled\_flag && !IsCuQpDeltaCoded ) { |  |
| **cu\_qp\_delta\_abs** | ae(v) |
| if( cu\_qp\_delta\_abs ) |  |
| **cu\_qp\_delta\_sign\_flag** | ae(v) |
| } |  |
| if( slice\_chroma\_qp\_adjustment\_enabled\_flag && cbfChroma && !IsCuChromaQpAdustmentCoded ) { |  |
| **cu\_chroma\_qp\_adustment\_flag** | ae(v) |
| if( cu\_chroma\_qp\_adjustment\_flag && chroma\_qp\_adjustment\_table\_size > 1 ) |  |
| **cu\_chroma\_qp\_adjustment\_idc** | ae(v) |
| } |  |
| ... |  |
| } |  |
| } |  |

#### Transform unit semantics

**cu\_chroma\_qp\_adjustment\_flag** equal to 1 specifies that a chroma qp adjustment shall be applied to the chroma qp offset during the scaling process of chroma transform coefficients. cu\_chroma\_qp\_adjustment\_flag equal to 0 specifies that no adjustment is applied to the chroma qp offset.

When cu\_chroma\_qp\_adjustment\_flag is present, the variable IsCuChromaQpAdustmentCoded is set equal to 1

– The variable qPa is derived as follows:

qPa = cu\_chroma\_qp\_adjustment\_flag ? chroma\_qp\_adjustment[ cu\_chroma\_qp\_adjustment\_idc ] : 0 (8‑263)

qPaCb = cu\_chroma\_qp\_adjustment\_flag ? cb\_qp\_adjustment[ cu\_chroma\_qp\_adjustment\_idc ] : 0 (8‑263)

qPaCr = cu\_chroma\_qp\_adjustment\_flag ? cr\_qp\_adjustment[ cu\_chroma\_qp\_adjustment\_idc ] : 0 (8‑263)

**cu\_chroma\_qp\_adjustment\_idc** specifies the index to chroma\_qp\_adjustment[ ] that is used in the derivation of Qp′Cb and Qp′Cr. When present, the value of cu\_chroma\_qp\_adjustment\_idc shall be in the range of 0 to chroma\_qp\_adjustment\_table\_size − 1 inclusive. When not present, the value of cu\_chroma\_qp\_adjustment\_idc shall be inferred to be equal to 0.

## Scaling, transformation and array construction process prior to deblocking filter process

### Derivation process for quantization parameters

...

When ChromaArrayType is not equal to 0, the following applies.

– The variables qPiCb and qPiCr are derived as follows:

qPiCb = Clip3( −QpBdOffsetC, 57, QpY + pps\_cb\_qp\_offset + slice\_cb\_qp\_offset + qPaCb ) (8‑263)

qPiCr = Clip3( −QpBdOffsetC, 57, QpY + pps\_cr\_qp\_offset + slice\_cr\_qp\_offset + qPaCr ) (8‑264)

– If ChromaArrayType is equal to 1, the variables qPCb and qPCr are set equal to the value of QpC as specified in Table 8‑10 based on the index qPiequal to qPiCb and qPiCr, respectively.

– Otherwise, the variables qPCb and qPCr are set equal to Min( qPi, 51 ), based on the index qPiequal to qPiCb and qPiCr, respectively.

– The chroma quantization parameters for the Cb and Cr components, Qp′Cb and Qp′Cr, are derived as follows:

Qp′Cb = qPCb + QpBdOffsetC (8‑265)

Qp′Cr = qPCr + QpBdOffsetC (8‑266)

#### Initialization process for context variables

Table 9‑4 – Association of ctxIdx and syntax elements for each initializationType in the initialization process

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Syntax structure** | **Syntax element** | **ctxTable** | **initType** | | |
| **0** | **1** | **2** |
| transform\_unit( ) | cu\_qp\_delta\_abs | Table 9‑24 | 0..1 | 2..3 | 4..5 |
| cu\_chroma\_qp\_adjustment\_flag | XXX | 0 | 1 | 2 |
| cu\_chroma\_qp\_adjustment\_idc | XXX | 0 | 1 | 2 |

Table 9‑5 – Values of initValue for ctxIdx of cu\_chroma\_qp\_adjustment\_flag and cu\_chroma\_qp\_adjustment\_idc

|  |  |  |  |
| --- | --- | --- | --- |
| **Initialization variable** | **ctxIdx of cu\_chroma\_qp\_adjustment\_flag** | | |
| **0** | **1** | **2** |
| **initValue** | 153 | 153 | 153 |

### Binarization process

#### General

| Table 9‑34 – Syntax elements and associated binarizations | | | |
| --- | --- | --- | --- |
| **Syntax structure** | **Syntax element** | **Binarization** | |
| **Process** | **Input parameters** |
| residual\_coding( ) | ... | ... | ... |
| cu\_chroma\_qp\_adjustment\_flag | FL | cMax = 1 |
| cu\_chroma\_qp\_adjustment\_idc | TR | cMax = chroma\_qp\_adjustment\_table\_size\_minus1, cRiceParam = 0 |

#### Derivation process for ctxTable, ctxIdx and bypassFlag

##### General

| Table 9‑39 – Assignment of ctxInc to syntax elements with context coded bins | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| **Syntax element** | **binIdx** | | | | | |
| **0** | **1** | **2** | **3** | **4** | **>= 5** |
| cu\_chroma\_qp\_adjustment\_flag | 0 | na | na | na | na | na |
| cu\_chroma\_qp\_adjustment\_idc | 0 | na | na | na | na | na |