The proposed text is added on the top of JCT3V-Gxxxx.

**H.7.3.8.5.1 Depth mode parameter syntax**

|  |  |
| --- | --- |
| depth\_mode\_parameters( x0 , y0 , log2CbSize ) { | **Descriptor** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| if ( DepthIntraMode[ x0 ][ y0 ] = = INTRA\_DEP\_DMM\_WFULL  ~~| |    DepthIntraMode[ x0 ][ y0 ] = = INTRA\_DEP\_SDC\_DMM\_WFULL~~ ) |  |
| **wedge\_full\_tab\_idx**[ x0  ][ y0 ] | ae(v) |
| if( DmmFlag[ x0 ][ y0 ]  | |  SdcFlag[ x0 ][ y0 ] ) { |  |
| dcNumSeg = ( DepthIntraMode[ x0 ][ y0 ] = = INTRA\_DEP\_SDC\_PLANAR ) ? 1 : 2 |  |
| **depth\_dc\_flag**[ x0 ][ y0 ] | ae(v) |
| if ( depth\_dc\_flag[ x0 ][ y0 ] ) |  |
| for( i = 0; i < dcNumSeg; i ++ ) { |  |
| **depth\_dc\_abs**[ x0 ][ y0 ][ i ] | ae(v) |
| if ( depth\_dc\_abs[ x0 ][ y0 ][ i ]) |  |
| **depth\_dc\_sign\_flag**[ x0 ][ y0 ][ i ] | ae(v) |
| } |  |
| } |  |
| } |  |

**H.7.3.8.5 Coding unit syntax**

|  |  |
| --- | --- |
| coding\_unit( x0, y0, log2CbSize , ctDepth) { |  |
| if( transquant\_bypass\_enable\_flag ) { |  |
| **cu\_transquant\_bypass\_flag** | ae(v) |
| … |  |
| } else { |  |
| pbOffset = ( PartMode = = PART\_NxN ) ? ( nCbS / 2 ) : 0 |  |
| log2PbSize = log2CbSize − ( PartMode = = PART\_NxN ? 1 : 0 ) |  |
| for( j = 0; j <= pbOffset; j = j + pbOffset ) |  |
| for( i = 0; i <= pbOffset; i = i + pbOffset ) { |  |
| if( vps\_depth\_modes\_flag[ nuh\_layer\_id ] ) { |  |
| depth\_mode\_parameters( x0 + i ,  y0+ j , log2CbSize ) |  |
| if ( log2CbSize < 6 ) |  |
| **hevc\_intra\_flag**[ x0 ][ y0 ] | ae(v) |
| if ( !hevc\_intra\_flag[ x0 ][ y0 ] && ( ( log2CbSize > 3 ) || ( log2CbSize = = 3 &&  PartMode[ xC ][ yC ] = = PART\_2Nx2N ) )  ) |  |
| **depth\_intra\_mode**[ x0 ][ y0 ] | ae(v) |
| } |  |
| ~~if( DepthIntraMode[ x0 + i ][ y0 + j ] = = INTRA\_DEP\_NONE )~~ |  |
| if( hevc\_intra\_flag[ x0 ][ y0 ]) |  |
| **prev\_intra\_luma\_pred\_flag**[ x0 + i ][ y0+ j ] | ae(v) |
| } |  |
| for( j = 0; j <= pbOffset; j = j + pbOffset ) |  |
| for( i = 0; i <= pbOffset; i = i + pbOffset ) { |  |
| ~~if( DepthIntraMode[ x0 + i ][ y0 + j ] = = INTRA\_DEP\_NONE) {~~ |  |
| if( hevc\_intra\_flag[ x0 ][ y0 ]) { |  |
| if( prev\_intra\_luma\_pred\_flag[ x0 + i ][ y0+ j ] ) |  |
| **mpm\_idx**[ x0 + i ][ y0+ j ] | ae(v) |
| else |  |
| **rem\_intra\_luma\_pred\_mode**[ x0 + i ][ y0+ j ] | ae(v) |
| } |  |
| } |  |
| if( vps\_depth\_modes\_flag[ nuh\_layer\_id ] ) { |  |
| for( j = 0; j <= pbOffset; j = j + pbOffset ) |  |
| for( i = 0; i <= pbOffset; i = i + pbOffset ) { |  |
| if ( PartMode[ xC ][ yC ] = = PART\_2Nx2N && (prev\_intra\_luma\_pred\_flag[ x0 ][ y0] || DmmFlag[x0][y0] ) |  |
| **sdc\_flag**[ x0 ][ y0 ] | ae(v) |
| depth\_mode\_parameters( x0 + i ,  y0+ j , log2CbSize ) |  |
| } |  |
| } |  |
| if ( !SdcFlag[ x0 ][ y0 ] ) |  |
| **intra\_chroma\_pred\_mode**[ x0 ][ y0 ] | ae(v) |
| } |  |
| **…** |  |
| if( !pcm\_flag ) { |  |
| if( PredMode[ x0 ][ y0 ] != MODE\_INTRA &&   !(PartMode = = PART\_2Nx2N && merge\_flag[x0][y0])) |  |
| **rqt\_root\_cbf** | ae(v) |
| if( rqt\_root\_cbf && !inter\_sdc\_flag ) { |  |
| MaxTrafoDepth = ( PredMode[ x0 ][ y0 ] = = MODE\_INTRA ?   max\_transform\_hierarchy\_depth\_intra + IntraSplitFlag :   max\_transform\_hierarchy\_depth\_inter ) |  |
| transform\_tree( x0, y0 x0, y0, log2CbSize, 0, 0 ) |  |
| } |  |
| } |  |
| } |  |
| … |  |

**Semantics and decoding process**

**H.7.4.9.5.1 Depth mode parameter semantics**

**hevc\_intra\_flag**[ x0 ][ y0 ] equal to 1 specifies that intra modes with intraPredMode in the range of 0 to 34 is used for the current prediction unit. hevc\_intra\_flag[ x0 ][ y0 ] equal to 0 specifies that another intra mode is used for the current prediction unit. When not present, hevc\_intra\_flag[ x0 ][ y0 ] is inferred to be equal to 1.

**sdc\_flag**[ x0 ][ y0 ] equal to 1 specifies that the residual of each partition of the current prediction unit is only represented by sdc\_residual\_flag[ x0 ][ y0 ][ i ], sdc\_residual\_sign\_flag [ x0 ][ y0 ][ i ] and sdc\_residual\_abs\_minus1[ x0 ][ y0 ][ i ]. sdc\_flag[ x0 ][ y0 ] equal to 0 specifies that more residual may be present in the transform tree. When not present, sdc\_flag[ x0 ][  y0 ] is inferred to be equal to 0.

The variable SdcFlag[ x0 ][ y0 ] is set to be equal to sdc\_flag[ x0 ][ y0 ].

The variable DmmFlag[ x0 ][ y0 ] is derived as specified in the following:

DmmFlag[ x0 ][ y0 ] = !hevc\_intra\_flag[ x0 ][ y0 ] (H‑25)

**depth\_intra\_mode**[ x0 ][ y0 ] specifies the depth intra mode of the current prediction unit. depth\_intra\_mode[ x0 ][ y0 ] equal to 0 specifies that DepthIntraMode[ x0 ][ y0 ] is equal to INTRA\_DEP\_DMM\_WFULL, and depth\_intra\_mode[ x0 ][ y0 ] equal to 1 specifies that DepthIntraMode[ x0 ][ y0 ] is equal to INTRA\_DEP\_DMM\_CPREDTEX. When not present, if hevc\_intra\_flag[ x0 ][ y0 ] is equal to 0, DepthIntraMode[ x0 ][ y0 ] is inferred to be equal to INTRA\_DEP\_DMM\_WFULL; otherwise, if hevc\_intra\_flag[ x0 ][ y0 ] is equal to 1, DepthIntraMode[ x0 ][ y0 ] is inferred to be equal to INTRA\_DEP\_NONE. ~~Table H‑2 specifies the value of the variable depthIntraModeMaxLen depending on depthIntraModeSet and the value of the variable DepthIntraMode and the associated name depending on the on depth\_intra\_mode and depthIntraModeSet.~~

~~The variable SdcFlag[ x0 ][ y0 ] is derived as specified in the following:~~

~~SdcFlag[ x0 ][ y0 ] = ( DepthIntraMode[ x0 ][ y0 ]  = =  INTRA\_DEP\_SDC\_PLANAR )  | |   (~~H~~‑25)  
 ( DepthIntraMode[ x0 ][ y0 ]  = =  INTRA\_DEP\_SDC\_DMM\_WFULL )~~

~~The variable DmmFlag[ x0 ][ y0 ] is derived as specified in the following:~~

~~DmmFlag[ x0 ][ y0 ] = ( DepthIntraMode[ x0 ][ y0 ]  = =  INTRA\_DEP\_DMM\_WFULL )  | |   (~~H~~‑26)  
 ( DepthIntraMode[ x0 ][ y0 ]  = =  INTRA\_DEP\_DMM\_CPREDTEX )~~

1. **~~Table~~**H**~~‑2 – Specification of DepthIntraMode and associated name depending on depthIntraModeSet and depth\_intra\_mode and specification of and depthIntraModeMaxLen depending on depthIntraModeSet~~**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **~~depthIntraModeSet~~** | ~~0~~ | ~~1~~ | ~~2~~ |  |
|  | **~~depthIntraModeMaxLen~~** | ~~1~~ | ~~3~~ | ~~3~~ |  |
| **~~DepthIntraMode~~** | **~~Associated name~~** | **~~depth\_intra\_mode~~** | | | |
| ~~0~~ | ~~INTRA\_DEP\_SDC\_PLANAR~~ | ~~0~~ | ~~-~~ | ~~0~~ |  |
| ~~1~~ | ~~INTRA\_DEP\_NONE~~ | ~~1~~ | ~~0~~ | ~~1~~ |  |
| ~~2~~ | ~~INTRA\_DEP\_SDC\_DMM\_WFULL~~ | ~~-~~ | ~~-~~ | ~~2~~ |  |
| ~~3~~ | ~~INTRA\_DEP\_DMM\_WFULL~~ | ~~-~~ | ~~1~~ | ~~3~~ |  |
| ~~4~~ | ~~INTRA\_DEP\_DMM\_CPREDTEX~~ | ~~-~~ | ~~-~~ | ~~4~~ |  |

…

**H.8.4.2 Derivation process for luma intra prediction mode**

Input to this process is a luma location ( xPb, yPb ) specifying the top-left sample of the current luma prediction block relative to the top left luma sample of the current picture.

In this process, the luma intra prediction mode IntraPredModeY[ xPb ][ yPb ] is derived.

Table H‑3 specifies the value for the intra prediction mode and the associated names.

**Table**H**‑3 – Specification of intra prediction mode and associated names**

|  |  |
| --- | --- |
| **Intra prediction mode** | **Associated name** |
| 0 | INTRA\_PLANAR |
| 1 | INTRA\_DC |
| 2..34 | INTRA\_ANGULAR2..INTRA\_ANGULAR34 |
| 35 | INTRA\_DMM\_WFULL |
| 36 | INTRA\_DMM\_CPREDTEX |

IntraPredModeY[ xPb ][ yPb ] labelled 0..34 represents directions of predictions as illustrated in Figure 8 1.

~~If DepthIntraMode[ xPb ][ yPb ] is equal to INTRA\_DEP\_SDC\_PLANAR, IntraPredModeY[ xPb ][ yPb ] is set equal to INTRA\_PLANAR.~~

~~Otherwise, if DepthIntraMode[ xPb ][ yPb ] is equal to INTRA\_DEP\_SDC\_DMM\_WFULL, IntraPredModeY[ xPb ][ yPb ] is set equal to INTRA\_DMM\_WFULL.~~

~~Otherwise, i~~If DepthIntraMode[ xPb ][ yPb ] is equal to INTRA\_DEP\_DMM\_WFULL, IntraPredModeY[ xPb ][ yPb ] is set equal to INTRA\_DMM\_WFULL.

Otherwise if DepthIntraMode[ xPb ][ yPb ] is equal to INTRA\_DEP\_DMM\_CPREDTEX, IntraPredModeY[ xPb ][ yPb ] is set equal to INTRA\_DMM\_CPREDTEX.

Otherwise ( DepthIntraMode[ xPb ][ yPb ] is equal to INTRA\_DEP\_NONE ), IntraPredModeY[ xPb ][ yPb ] is derived as the following ordered steps:

…

**H.8.4.4.2.1 General intra sample prediction**

Inputs to this process are:

a sample location ( xTbCmp, yTbCmp ) specifying the top-left sample of the current transform block relative to the top left sample of the current picture,

a variable predModeIntra specifying the intra prediction mode,

a variable nTbS specifying the transform block size,

a variable cIdx specifying the colour component of the current block.

Output of this process is the predicted samples predSamples[ x ][ y ], with x, y = 0..nTbS − 1.

Set variable bSamplePredFlag to 1.

...

Otherwise, if predModeIntra is equal to INTRA\_DMM\_CPREDTEX, the corresponding intra prediction mode specified in subclause H.8.4.4.2.8 is invoked with the location ( xTbY, yTbY ), with the sample array p and the transform block size nTbS and bSamplePredFlag as the inputs and the output are the predicted sample array predSamples.

**H.8.4.4.2.8 Specification of intra prediction mode INTRA\_DMM\_CPREDTEX**

Inputs to this process are:

a sample location ( xTb, yTb ) specifying the top-left sample of the current block relative to the top‑left sample of the current picture,

the neighbouring samples p[ x ][ y ], with x = −1, y = −1..nTbS \* 2 − 1 and x = 0..nTbS \* 2 − 1, y = −1,

a variable nTbS specifying the transform block size,

a variable bSamplePredFlag specifying whether to generate the prediction samples or not,

Output of this process is:

the predicted samples predSamples[ x ][ y ], with x, y = 0..nTbS − 1.

wedgelet pattern wedgePattern[ x ][ y ], with x, y = 0..nT−1.

The values of the prediction samples predSamples[ x ][ y ], with x, y = 0..nTbS − 1, are derived as specified by the following ordered steps:

* 1. The variable recTextPic is set equal to the array of the reconstructed luma picture samples of TexturePic..
  2. The variable textThresh specifying a threshold for the segmentation of recTextPic is derived as specified in the following.
     + The variable sumTextPicVals is set equal to 0.
     + For x = 0..nTbS − 1 the following applies
       - For y = 0..nTbS − 1 the following applies

sumTextPicVals += recTextPic[ xTb + x ][ yTb + y ] (H‑45)

* + - The variable textThresh is set equal to ( sumTextPicVals >> ( 2 \* log2( nTbS ) ) )
  1. The variable wedgeletPattern[ x ][ y ] with x, y =0..nTbS − 1 specifying a binary partition pattern is derived as specified in the following.
     + For x = 0..nTbS − 1 the following applies
       - For y = 0..nTbS − 1 the following applies

wedgeletPattern[ x ][ y ] = ( recTextPic[ xTb + x ][ yTb + y ] > textThresh ) (H‑46)

* 1. When bSamplePredFlag is equal to 1, The depth partition value derivation and assignment process as specified in subclause H.8.4.4.2.9 is invoked with the neighbouring samples p[ x ][ y ], the binary pattern wedgeletPattern[ x ][ y ], the transform size nT, the dcOffsetAvailFlag set equal to depth\_dc\_flag[ xTb ][ yTb ], and the DC Offsets DcOffset[ xTb ][ yTb ][ 0 ], and DcOffset[ xTb ][ yTb ][ 1 ] as inputs and the output is assigned to predSamples[ x ][ y ].

**H.8.4.4.3 Depth value reconstruction process**

Inputs to this process are:

* a luma location ( xTb, yTb ) specifying the top-left luma sample of the current block relative to the top-left luma sample of the current picture,
* a variable nTbS specifying the transform block size,
* predicted samples predSamples[ x ][ y ], with x, y =0..nTbS − 1
* the intra prediction mode predModeIntra,

Output of this process is:

* reconstructed depth value samples resSamples[ x ][ y ], with x, y = 0.. nTbS − 1.

1. Set variable bSamplePredFlag to 0.

Depending on predModeIntra the array wedgePattern[ x ][ y ] with x, y =0..nTbS − 1 specifying the binary segmentation pattern is derived as follows.

* If predModeIntra is equal to INTRA\_DMM\_WFULL, the following applies.

wedgePattern = WedgePatternTable[ Log2( nTbS ) ][ wedge\_full\_tab\_idx[ xTb ][ yTb ] ]

* Otherwise, if predModeIntra is equal to INTRA\_DMM\_CPREDTEX, subclause H.8.4.4.2.8 is invoked with the location ( xB, yB ), predicted samples predSamples, the transform block size nT and bSamplePredFlag as the inputs and the output is the wedgelet pattern wedgePattern.
* Otherwise (predModeIntra is not equal to INTRA\_DMM\_WFULL and predModeIntra is not equal to INTRA\_DMM\_CPREDTEX), the following applies.
  + For x, y = 0..nTbS − 1 wedgePattern[ x ][ y ] is set equal to 0.

Depending on dlt\_flag[ nuh\_layer\_id ] the reconstructed depth value samples resSamples[ x ][ y ] are derived as specified in the following:

* If dlt\_flag[ nuh\_layer\_id ] is equal to 0, the following applies:
  + For x, y = 0..nTbS − 1, the reconstructed depth value samples resSamples[ x ][ y ] are derived as specified in the following:

resSamples[ x ][ y ] = predSamples[ x ][ y ] + DcOffset[ xTb ][ yTb ][wedgePattern[ x ][ y ] ] (H‑59)

* Otherwise ( dlt\_flag[ nuh\_layer\_id ] is equal to 1 ), the following applies:
  + The variables dcPred[ 0 ] and dcPred[ 1 ] are derived as specified in the following:
    - ~~If predModeIntra is equal to INTRA\_DC, the following applies:~~

~~dcPred[ 0 ] = predSamples[ nTbS − 1 ][ nTbS − 1 ] (~~H~~‑60)~~

* + - ~~Otherwise, if predModeIntra is equal to INTRA\_PLANAR~~ If predModeIntra is not equal to INTRA\_DMM\_WFULL and predModeIntra is not equal to INTRA\_DMM\_CPREDTEX, the following applies:

dcPred[ 0 ] = ( predSamples[ 0 ][ 0 ] + predSamples[ 0 ][ nTbS − 1 ] + predSamples[ nTbS − 1 ][ 0 ]  
 + predSamples[ nTbS − 1 ][ nTbS − 1 ] + 2 ) >> 2 (H‑61)

* + - Otherwise, ~~( predModeIntra is equal to INTRA\_DMM\_WFULL )~~if predModeIntra is equal to INTRA\_DMM\_WFULL, the following applies.

dcPred[ wedgePattern[ 0 ][ 0 ] ] = predSamples[ 0 ][ 0 ] (H‑62)

dcPred[ wedgePattern[ nTbS − 1 ][ 0 ] ] = predSamples[ nTbS − 1 ][ 0 ] (H‑63)

dcPred[ wedgePattern[ 0 ][ nTbS − 1 ] ] = predSamples[ 0 ][ nTbS − 1 ] (H‑64)

dcPred[ wedgePattern[ nTbS − 1 ][ nTbS − 1 ] ] = predSamples[ nTbS − 1 ][ nTbS − 1 ] (H‑65)

* + - Otherwise, (intraPredMode is equal to INTRA\_DMM\_CPREDTEX), the following applies.

dcPred[ wedgePattern[0][0] ] = predSamples[ 0 ][ 0 ]

dcPred1Found = 0;

for ( x = 0; x < nT – 1; x++ )  
 for ( y = 0; y < nT – 1; y++ )  
 if ( wedgePattern[ x ][ y ] != wedgePattern[ 0 ][ 0 ] && dcPred1Found == 0 ) {  
 dcPred[ wedgePattern[ x ][ y ] ] = predSamples[ x ][ y ]  
 dcPred1Found = 1  
 }

* + For x, y = 0..nTbS − 1, the reconstructed depth value samples resSamples[ x ][ y ] are derived as specified in the following:

dltIdxPred = DepthValue2Idx[ dcPred[ wedgePattern[ x ][ y ] ] ] (H‑66)

dltIdxResi = DcOffset[ xTb ][ yTb ][wedgePattern[ x ][ y ] ] (H‑67)

resSamples[ x ][ y ] = predSamples[ x ][ y ] + Idx2DepthValue[ dltIdxPred + dltIdxResi ]  −   
 dcPred[ wedgePattern[ x ][ y ] ] (H‑68)

**H.9.3.2.2 Initialization process for context variables**

…

**Table H‑10 – Association of ctxIdx and syntax elements for each initializationType in the initialization process**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Syntax structure** | **Syntax element** | **ctxIdxTable** | **initType** | | |
| **0** | **1** | **2** |
| coding\_unit( ) depth\_mode\_parameters( ) | … |  |  |  |  |
| sdc\_flag | Table H‑24 | 0..2 | 3..5 | 6..8 |
| hevc\_intra\_flag | Table H‑25 | 0..2 | 3..5 | 6..8 |
| ... |  |  |  |  |

…

**Table**H**‑20 – Values of variable initValue for sdc\_flag ctxIdx**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Initialization variable** | **ctxIdx of sdc\_flag** | | | | | | | | |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| **initValue** | 214 | 229 | 230 | 215 | 202 | 174 | 213 | 201 | 246 |

**Alternatively, 1 context is used for CABAC coding of sdc\_flag, Table H-24 is designed as follows.**

**Table**H**‑21 – Values of variable initValue for hevc\_intra\_flag ctxIdx**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Initialization variable** | **ctxIdx of hevc\_intra\_flag** | | | | | | | | |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| **initValue** | 154 | 155 | 156 | 141 | 185 | 214 | 155 | 170 | 157 |

…

H.9.3.3.1 General

The specifications in subclause 9.3.3.1 apply with the following modifications.

* Table H‑20 is appended to the end of Table 9-32.

**Table**H**‑20 – Syntax elements and associated binarizations**

| **Syntax structure** | **Syntax element** | **Binarization** | |
| --- | --- | --- | --- |
| **Process** | **Input parameters** |
| depth\_mode\_parameters( ) | depth\_intra\_mode | TR | cMax = depthIntraModeMaxLen, cRiceParam = 0 |
| wedge\_full\_tab\_idx | FL | cMax = wedgeFullTabIdxBits[ log2PbSize ] (defined in Table H‑21) |
| depth\_dc\_flag | FL | cMax = 1 |
| depth\_dc\_abs | UEG0 |  |
| depth\_dc\_sign\_flag | FL | cMax = 1 |
| sdc\_flag | FL | cMax = 1 |
| hevc\_intra\_flag | FL | cMax = 1 |

**H.9.3.4.2.2 Derivation process of ctxInc using left and above syntax elements**

The specifications in subclause 9.3.4.2.2 apply with the following modifications.

– Table H‑23 is appended to the end of Table 9-38.

**Table H‑23 – Specification of ctxInc using left and above syntax elements**

|  |  |  |  |
| --- | --- | --- | --- |
| **Syntax element** | **condL** | **condA** | **ctxIdxInc** |
| iv\_res\_pred\_weight\_idx | iv\_res\_pred\_weight\_idx [ xL ][ yL ] | iv\_res\_pred\_weight\_idx [ xA ][ yA ] | ( condL  &&  availableL ) + ( condA  &&  availableA ) |
| ic\_flag | ic\_flag[ xL ][ yL ] | ic\_flag[ xA ][ yA ] | ( condL  &&  availableL ) + ( condA  &&  availableA ) |
| sdc\_flag | sdc\_flag[ xL ][ yL ] | sdc\_flag[ xA ][ yA ] | ( condL  &&  availableL ) + ( condA  &&  availableA ) |
| hevc\_intra\_flag | hevc\_intra\_flag [ xL ][ yL ] | hevc\_intra\_flag [ xA ][ yA ] | ( condL  &&  availableL ) + ( condA  &&  availableA ) |