H.7.3.9.3 Depth mode parameter syntax

|  |  |
| --- | --- |
| depth\_mode\_parameters( x0 , y0 ) { | **Descriptor** |
| **depth\_intra\_mode**[ x0 ][ y0 ] | ae(v) |
| if ( depth\_intra\_mode[ x0 ][ y0 ] = = INTRA\_DEP\_DMM\_WFULL  | |    depth\_intra\_mode[ x0 ][ y0 ] = = INTRA\_DEP\_SDC\_DMM\_WFULL ) |  |
| **wedge\_full\_tab\_idx**[ x0  ][ y0 ] | ae(v) |
|  |  |
|  | ae(v) |
|  |  |
|  | ae(v) |
|  |  |
|  | ae(v) |
|  | ae(v) |
|  |  |
| - else if( depth\_intra\_mode[ x0 ][ y0 ] = = INTRA\_DEP\_CHAIN ) { |  |
| **edge\_start\_left\_flag**[ x0 ][ y0 ] | ae(v) |
| **edge\_start\_position**[ x0 ][ y0 ] | ae(v) |
| **num\_edge\_codes\_minus1**[ x0 ][ y0 ] | ae(v) |
| for( k = 0; k <= num\_edge\_codes\_minus1; k++ ) |  |
| **edge\_code**[ k ] | ae(v) |
| } |  |
| if( DmmFlag[ x0 ][ y0 ] ) { |  |
| **dmm\_dc\_flag**[ x0 ][ y0 ] | ae(v) |
| if ( dmm\_dc\_flag[ x0 ][ y0 ] ) |  |
| for( i = 0; i < 2; i ++ ) { |  |
| **dmm\_dc\_abs**[ x0 ][ y0 ][ i ] | ae(v) |
| if ( dmm\_dc\_abs[ x0 ][ y0 ][ i ]) |  |
| **dmm\_dc\_sign\_flag**[ x0 ][ y0 ][ i ] | ae(v) |
| } |  |
| } |  |
| else if( depth\_intra\_mode[ x0 ][ y0 ] = = INTRA\_DEP\_CHAIN ) { |  |
| **edge\_dc\_flag**[ x0 ][ y0 ] | ae(v) |
| if( edge\_dc\_flag[ x0 ][ y0 ] ) { |  |
| for( i = 0; i < 2; i++ ) { |  |
| **edge\_dc\_abs**[ x0 ][ y0 ][ i ] | ae(v) |
| if( edge\_dc\_abs[ x0 ][ y0 ][ i ] != 0 ) |  |
| **edge\_dc\_sign\_flag**[ x0 ][ y0 ][ i ] | ae(v) |
| } |  |
| } |  |
| } |  |
| else if( SdcFlag[ x0 ][ y0 ] ) { |  |
| sdcNumSegments =   ( depth\_intra\_mode[ x0  ][ y0 ] = = INTRA\_DEP\_SDC\_DMM\_WFULL ) ? 2 : 1 |  |
| for( i = 0; i < sdcNumSegments; i++ ) { |  |
| **sdc\_residual\_flag**[ x0 ][ y0 ][ i ] | ae(v) |
| if( sdc\_residual\_flag[ x0 ][ y0 ][ i ])  { |  |
| **sdc\_residual\_sign\_flag**[ x0 ][ y0 ][ i ] | * + 1. ae(v) |
| **sdc\_residual\_abs\_minus1**[ x0 ][ y0 ][ i ] | * + 1. ae(v) |
| } |  |
| } |  |
| } |  |
| } |  |

H.7.4.9.3 Depth mode parameter semantics

The variable Log2MaxDmmCbSize is set equal to 5.

**depth\_intra\_mode**[ x0 ][ y0 ] specifies the depth intra mode of the current prediction unit. Table H‑2 specifies the value for the depth intra mode and the associated names.

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

* 1. SdcFlag[ x0 ][ y0 ] = ( depth\_intra\_mode[ x0 ][ y0 ]  = =  INTRA\_DEP\_SDC\_PLANAR )  | |   (H‑16)  
      ( depth\_intra\_mode[ x0 ][ y0 ]  = =  INTRA\_DEP\_SDC\_DMM\_WFULL )  | |    
      ( depth\_intra\_mode[ x0 ][ y0 ]  = =  INTRA\_DEP\_SDC\_DC )

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

* 1. DmmFlag[ x0 ][ y0 ] = ( depth\_intra\_mode[ x0 ][ y0 ]  = =  INTRA\_DEP\_DMM\_WFULL )     (H‑17)

Table ‑2 – Specification of depth\_intra\_mode and associated names

|  |  |
| --- | --- |
| **depth\_intra\_mode** | **DepthIntraMode** |
| 0 | INTRA\_DEP\_SDC\_PLANAR |
| 1 | INTRA\_DEP\_NONE |
| 2 | INTRA\_DEP\_SDC\_DMM\_WFULL |
| 3 | INTRA\_DEP\_DMM\_WFULL |
|  |  |
|  |  |
| 4 | INTRA\_DEP\_SDC\_DC |
|  |  |
| 5 | INTRA\_DEP\_CHAIN |

**wedge\_full\_tab\_idx**[ x0 ][ y0 ]specifies the index of the wedgelet pattern in the corresponding pattern list when depth\_intra\_mode[ x0 ][ y0 ] is equal to INTRA\_DEP\_DMM\_WFULL.

* 1. (H‑18)

**edge\_start\_left\_flag**[ x0 ][ y0 ] equal to 0 specifies that the start point of region boundary chain coding is located on the top row boundary of the current block. edge\_start\_left\_flag[ x0 ][ y0 ] equal to 1 specifies that the start point of region boundary chain coding is located on the left column boundary of the current block.

**edge\_start\_position**[ x0 ][ y0 ] specifies the column position of the start point of region boundary chain coding when edge\_start\_left\_flag[ x0 ][ y0 ] is equal to 0 and specifies the row position of the start point of region boundary chain coding when edge\_start\_left\_flag[ x0 ][ y0 ] is equal to 1.

**num\_edge\_codes\_minus1**[ x0 ][ y0 ] +1 specifies the number of edges within the current block.

**edge\_code**[ x0 ][ y0 ][ k ] shall be one of the values shown in . edge\_code[ x0 ][ y0 ][ k ] is used to derive the edge direction of the k-th edge when edge\_intra\_flag[ x0 ][ y0 ] is equal to 1.

Table H‑3 – Interpretation of edge\_code[ x0 ][ y0 ][ k ]

|  |  |
| --- | --- |
| **edge\_code**[ x0 ][ y0 ]**[**k **]** | **edge direction** |
| 0 | 0° |
| 1 | 45° |
| 2 | −45° |
| 3 | 90° |
| 4 | −90° |
| 5 | 135° |
| 6 | −135° |

**dmm\_dc\_flag**[ x0 ][ y0 ] equal to 1 specifies that dmm\_dc\_abs[ x0 ][ y0 ][ i ] and dmm\_dc\_sign\_flag[ x0 ][ y0 ][ i ] are present, dmm\_dc\_flag[ x0 ][ y0 ] equal to 0 specifies that dmm\_dc\_abs[ x0 ][ y0 ][ i ] and dmm\_dc\_sign\_flag[ x0 ][ y0 ][ i ] are not present.

**dmm\_dc\_abs**[ x0][ y0 ][ i ], **dmm\_dc\_sign\_flag**[ x0 ][ y0 ][ i ]are used to derive DcOffset[ x0 ][ y0 ][ i ] as follows:

* 1. DcOffset[ x0 ][ y0 ][ i ] =   
     ( 1 − 2 \*dmm\_dc\_sign\_flag[ x0 ][ y0 ][ i ] ) \* dmm\_dc\_abs[ x0 ][ y0 ][ i ] (H‑19)

**edge\_dc\_flag**[ x0 ][ y0 ] equal to 1 specifies that edge\_dc\_abs[ x0 ][ y0 ][ i ] and edge\_dc\_sign\_flag[ x0 ][ y0 ][ i ] are present, edge\_dc\_flag[ x0 ][ y0 ] equal to 0 specifies that edge\_dc\_abs[ x0 ][ y0 ][ i ] and edge\_dc\_sign\_flag[ x0 ][ y0 ][ i ] are not present.

**edge\_dc\_abs**[ x0][ y0 ][ i ], **edge\_dc\_sign\_flag**[ x0 ][ y0 ][ i ]are used to derive DcOffset[ x0 ][ y0 ][ i ] as follows:

* 1. DcOffset[ x0 ][ y0 ][ i ] =   
     ( 1 − 2 \*edge\_dc\_sign\_flag[ x0 ][ y0 ][ i ] ) \* edge\_dc\_abs[ x0 ][ y0 ][ i ] (H‑20)

**sdc\_residual\_flag**[ x0 ][ y0 ][ i ] equal to 0 specifies that the residual is zero for segment i. sdc\_residual\_flag equal to 1 specifies that the residual is non-zero and the sdc\_residual\_sign\_flag and sdc\_residual\_abs[ i ] syntax elements are present for segment i.

**sdc\_residual\_abs\_minus1**[ x0 ][ y0 ][ i ]and **sdc\_residual\_sign\_flag**[ x0 ][ y0 ][ i ]are used to derive SdcResidual[ x0 ][ y0 ][ i ] for segment i as follows:

* 1. SdcResidual[ x0 ][ y0 ][i] =   
     ( 1 − 2 \*sdc\_residual\_sign\_flag[ x0 ][ y0 ][ i ] ) \* ( sdc\_residual\_mag\_minus1[ x0 ][ y0 ][ i ] + 1) (H‑21)

H.8.4.2 Derivation process for luma intra prediction mode

Inputs to this process are:

– a luma location ( xB, yB ) specifying the top-left luma sample of the current block relative to the top‑left luma sample of the current picture,

– a variable log2PbSize specifying the size of the current luma prediction block.

specifies the value for the intra prediction mode and the associated names.

Table H‑4 – Specification of intra prediction mode and associated names

|  |  |
| --- | --- |
| **Intra prediction mode** | **Associated names** |
| 0 | Intra\_Planar |
| 1 | Intra\_DC |
| 2..34 | Intra\_Angular |
| 35, 36 | Intra\_DepthPartition (used only for depth) |
| Otherwise (37, 38) | Intra\_Chain (used only for depth) |

[Ed. (GT): Since the Intra\_FromLuma mode has been removed in the HEVC text spec used as base for this document, the dmm mode and intra chain numbers are here decremented by 1 compared to HTM-7.0.]

[Ed. (GT): Consider reducing number of possible IntraPredMode values by using dmm\_dc\_flag and edge\_dc\_flag explicitly. ]

IntraPredMode[ xB ][ yB ] labelled 0..34 represents directions of predictions as illustrated in Figure 8-1.

* If depth\_intra\_mode[ xB ][ yB ] is equal to INTRA\_DEP\_SDC\_PLANAR, IntraPredMode[ xB ][ yB ] is set equal to Intra\_Planar.
* Otherwise, if depth\_intra\_mode[ xB ][ yB ] is equal to INTRA\_DEP\_SDC\_DMM\_WFULL, IntraPredMode[ xB ][ yB ] is set equal to Intra\_DepthPartition( 35 ).
* Otherwise if sdc\_pred\_mode[ xB ][ yB ] is equal to INTRA\_DEP\_SDC\_DC, IntraPredMode[ xB ][ yB ] is set equal to Intra\_DC.
* Otherwise, if depth\_intra\_mode[ xB ][ yB ] is equal to INTRA\_DEP\_DMM\_WFULL, IntraPredMode[ xB ][ yB ] is set equal to Intra\_DepthPartition( 35 + dmm\_dc\_flag[ xB ][ yB ] ) .
* Otherwise if depth\_intra\_mode[ xB ][ yB ] is equal to INTRA\_DEP\_CHAIN, IntraPredMode[ xB ][ yB ] is set equal to Intra\_Chain( 37 + edge\_dc\_flag[ xB ][ yB ] ).
* Otherwise ( depth\_intra\_mode[ xB ][ yB ] is equal to INTRA\_DEP\_NONE ), IntraPredMode[ xB ][ yB ] is derived as the following ordered steps.
  1. The neighbouring locations ( xBA, yBA ) and ( xBB, yBB ) are set equal to ( xB−1, yB ) and ( xB, yB−1 ), respectively.
  2. For N being either replaced A or B, the variables candIntraPredModeN are derived as follows.
     + The availability derivation process for a block in z-scan order as specified in subclause 6.4.1 is invoked with the location ( xCurr, yCurr ) set equal to ( xB, yB ) and the neighbouring location ( xN, yN ) set equal to ( xBN, yBN ) as the input and the output is assigned to availableN.
     + The candidate intra prediction mode candIntraPredModeN is derived as follows.
       - If availableN is equal to FALSE, candIntraPredModeN is set equal to Intra\_DC.
       - Otherwise, if PredMode[ xBN ][ yBN ] is not equal to MODE\_INTRA, candIntraPredModeN is set equal to Intra\_DC,
       - Otherwise, if N is equal to B and yB−1 is less than (( yB >> Log2CtbSizeY ) << Log2CtbSizeY), intraPredModeB is set equal to Intra\_DC.
       - Otherwise, if candIntraPredModeN is larger than 34, candIntraPredModeN is set equal to Intra\_DC.
       - Otherwise, candIntraPredModeN is set equal to IntraPredMode[ xBN ][ yBN ].
  3. The candModeList[ x ] with x=0..2 is derived as follows:
     + If candIntraPredModeB is equal to candIntraPredModeA, the following applies:
       - If candIntraPredModeA is less than 2 (either Intra\_Planar or Intra\_DC), candModeList[ x ] with x=0..2 is derived as:

candModeList[0] = Intra\_Planar (H‑22)  
candModeList[1] = Intra\_DC (H‑23)  
candModeList[2] = Intra\_Angular (26) (H‑24)

* + - * Otherwise, candModeList[ x ] with x=0..2 is derived as:

candModeList[0] = candIntraPredModeA (H‑25)  
candModeList[1] = 2 + ( ( candIntraPredModeA + 29 ) % 32 ) (H‑26)  
candModeList[2] = 2 + ( ( candIntraPredModeA − 2 + 1 ) % 32 ) (H‑27)

* + - Otherwise (candIntraPredModeB is not equal to candIntraPredModeA), the following applies:
      * candModeList[0] and candModeList[1] are derived as follows:

candModeList[0] = candIntraPredModeA (H‑28)  
candModeList[1] = candIntraPredModeB (H‑29)

* + - * If none of candModeList[0] and candModeList[1] is equal to Intra\_Planar, candModeList[2] is set equal to Intra\_Planar,
      * Otherwise, if none of candModeList[0] and candModeList[1] is equal to Intra\_DC, candModeList[2] is set equal to Intra\_DC,
      * Otherwise, candModeList[2] is set equal to Intra\_Angular (26).
  1. IntraPredMode[ xB ][ yB ] is derived by applying the following procedure:
     + If prev\_intra\_luma\_pred\_flag[ xB ][ yB ] is equal to 1, the IntraPredMode[ xB ][ yB ] is set equal to candModeList[ mpm\_idx ].
     + Otherwise IntraPredMode[ xB ][ yB ] is derived by applying the following ordered steps:

1. The array candModeList[x], x=0..2 is modified as the following ordered steps:
   1. When candModeList[0] is greater than candModeList[1], both values are swapped as follows.

( candModeList[0], candModeList[1] ) = Swap( candModeList[0], candModeList[1] ) (H‑30)

* 1. When candModeList[0] is greater than candModeList[2], both values are swapped as follows.

( candModeList[0], candModeList[2] ) = Swap( candModeList[0], candModeList[2] ) (H‑31)

* 1. When candModeList[1] is greater than candModeList[2], both values are swapped as follows.

( candModeList[1], candModeList[2] ) = Swap( candModeList[1], candModeList[2] ) (H‑32)

1. IntraPredMode[xB][yB] is derived as the following ordered steps:
   1. IntraPredMode[ xB ][ yB ] = rem\_intra\_luma\_pred\_mode[ xB ][ yB ]
   2. When IntraPredMode[ xB ][ yB ] is greater than or equal to candModeList[ 0 ], the value of IntraPredMode[ xB ][ yB ] is increased by one
   3. When IntraPredMode[ xB ][ yB ] is greater than or equal to candModeList[ 1 ], the value of IntraPredMode[ xB ][ yB ] is increased by one

When IntraPredMode[ xB ][ yB ] is greater than or equal to candModeList[ 2 ], the value of IntraPredMode[ xB ][ yB ] is increased by one

H.8.4.4.2 Intra sample prediction

H.8.4.4.2.1 General intra sample prediction

The specification in subclause 8.4.4.2.1 with the following paragraphs added to the end of subclause apply:

* Otherwise, if intraPredMode is equal to Intra\_DepthPartition(35,36), the corresponding intra prediction mode specified in subclause H.8.4.4.2.7 is invoked with the location ( xB0, yB0 ), the intra prediction mode intraPredMode, the sample array p and the transform block size nT as the inputs and the output are the predicted sample array predSamples.
* Otherwise, if intraPredMode is equal to Intra\_Chain(37,38), the corresponding intra prediction mode specified in subclause 8 is invoked with the location ( xB0, yB0 ), the intra prediction mode intraPredMode, with the sample array p and the transform block size nT as the inputs and the output are the predicted sample array predSamples.

H.8.4.4.2.7 Specification of Intra\_DepthPartition (35, 36) prediction mode

Inputs to this process are:

* a sample location ( xB, yB ) specifying the top-left sample of the current block relative to the top‑left sample of the current picture,
* a variable intraPredMode specifying the intra prediction mode,
* the neighbouring samples p[ x ][ y ], with x, y = −1..2\*nT−1,
* a variable nT specifying the transform size,

Output of this process is:

* predicted samples predSamples[ x ][ y ], with x, y =0..nT−1.

This intra prediction mode is invoked when intraPredMode is equal to 35 or 36.

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

* 1. The variable wedgePattern[ x ][ y ] with x, y =0..nT−1, specifying a binary partition pattern is derived as.
     + 1. wedgePattern = WedgePatternTable[ Log2( nT) ][ wedge\_full\_tab\_idx[ xB ][ yB ] ] (H‑33)
  2. The depth partition value derivation and assignment process as specified in subclause H.8.4.4.2.12 is invoked with the neighbouring samples p[ x ][ y ], the binary pattern wedgePattern [ xB ][ yB ], the transform size nT, the dcOffsetAvailFlag set equal to ( intraPredMode = = Intra\_DepthPartition(36) ), intraChainFlag set equal to 0, and the DC Offsets DcOffset[ xB ][ yB ][ 0 ], and DcOffset[ xB ][ yB ][ 1 ] as inputs and the output is assigned to predSamples[ x ][ y ].
  3. For x, y = 0..nT − 1, inclusive the following applies:
     + WedgeIdx[ xB +x ][ yB +y] is set equal to wedge\_full\_tab\_idx[ xB ][ yB ].







H.8.4.4.2.11 Specification of Intra\_Chain (37, 38) prediction mode

Inputs to this process are:

* a sample location ( xB, yB ) specifying the top-left sample of the current block relative to the top‑left sample of the current picture,
* a variable intraPredMode specifying the intra prediction mode,
* the neighbouring samples p[ x ][ y ], with x, y = −1..2\*nT−1,
* a variable nT specifying the transform size,

Output of this process is:

* predicted samples predSamples[ x ][ y ], with x, y = 0..nT−1.

This intra prediction mode is invoked when intraPredMode is equal to 43 or 44.

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

* 1. [Ed. (GT) The process of edge reconstruction (items 1 - 3) need to be specified more in detail. In current state the specification is not precise enough. (#3)]
  2. The start point of the chain in the current prediction unit is derived from edge\_start\_left\_flag[ xB ][ yB ] and edge\_start\_position[ xB ][ yB ].
* If edge\_start\_left\_flag[ xB ][ yB ] is equal to 1, the start point is set as s[ 0 ][ yS ] where yS is equal to edge\_start\_position[ xB ][ yB ].
* Otherwise, the start point is set as s[ xS ][ 0 ] where xS is equal to edge\_start\_position[ xB ][ yB ].
  1. The number of edges and the direction of each edge are derived from num\_edge\_codes\_minus1[ xB ][ yB ] and edge\_code[ xB ][ yB ][ k ], for k = 0.. num\_edge\_codes\_minus1[ xB ][ yB ]. The direction of the edge is derived from edge\_code as shown in .
  2. The region boundary generated by connecting each edge separates the predicted samples predSamples[ x ][ y ], with x, y = 0..nT−1 into two regions: the region rA that covers the top-left pixel (x0, y0) and the region rB that covers the remaining region.

For x, y = 0..nT−1 the binary pattern edgePattern[ x ][ y ] is derived as follows:

* If (x, y) is covered by rA, edgePattern[ x ][ y ] = 0.
* Otherwise, edgePattern[ x ][ y ] = 1.
  1. The depth partition value derivation and assignment process as specified in subclause H.8.4.4.2.12 is invoked with the neighbouring samples p[ x ][ y ], the binary pattern edgePattern[ x ][ y ], the transform size nT, the dcOffsetAvailFlag set equal to ( intraPredMode = = Intra\_DepthPartition(44) ), intraChainFlag set equal to 1, and the DC offsets DcOffset[ xB ][ yB ][ 0 ], and DcOffset[ xB ][ yB ][ 1 ] as inputs and the output is assigned to predSamples[ x ][ y ].

Table H‑12 – Association of ctxIdx and syntax elements for each initializationType in the initialization process

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Syntax element** | **ctxIdxTable** | **initType** | | |
| **0** | **1** | **2** |
| coding\_unit( ) depth\_mode\_parameters( ) | depth\_intra\_mode |  | 0..7 | 8..15 | 16..23 |
| wedge\_full\_tab\_idx |  | 0 | 1 | 2 |
|  |  |  |  |  |
|  |  |  |  |  |
| dmm\_dc\_flag |  | 0 | 1 | 2 |
| dmm\_dc\_abs |  | 0 | 1 | 2 |
| edge\_code |  | 0 | 1 | 2 |
| edge\_dc\_flag |  | 0 | 1 | 2 |
| edge\_dc\_abs |  | 0 | 1 | 2 |

Table H‑13 – Values of variable initValue for wedge\_full\_tab\_idx ctxIdx

|  |  |  |  |
| --- | --- | --- | --- |
| **Initialization variable** | **wedge\_full\_tab\_idx** | | |
| **0** | **1** | **2** |
| **initValue** | 154 | 154 | 154 |







Table H‑16 – Values of variable initValue for dmm\_dc\_abs ctxIdx

|  |  |  |  |
| --- | --- | --- | --- |
| **Initialization variable** | **dmm\_dc\_abs** | | |
| **0** | **1** | **2** |
| **initValue** | 154 | 154 | 154 |

Table H‑17 – Values of variable initValue for edge\_code ctxIdx

|  |  |  |  |
| --- | --- | --- | --- |
| **Initialization variable** | **edge\_code** | | |
| **0** | **1** | **2** |
| **initValue** | 154 | 154 | 154 |

Table H‑22 – Values of variable initValue for depth\_intra\_mode ctxIdx

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Initialization variable** | **depth\_intra\_mode** | | | | | | | |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| **initValue** | 0 | 0 | 64 | 0 | 154 | 0 | 154 | 0 |
|  | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |
| **initValue** | 0 | 64 | 0 | 154 | 0 | 154 | 0 | 0 |
|  | **16** | **17** | **18** | **19** | **20** | **21** | **22** | **23** |
| **initValue** | 64 | 0 | 154 | 0 | 154 | 0 | 0 | 0 |

Table H‑23 – Values of variable initValue for dmm\_dc\_flag ctxIdx

|  |  |  |  |
| --- | --- | --- | --- |
| **Initialization variable** | **dmm\_dc\_flag** | | |
| **0** | **1** | **2** |
| **initValue** | 0 | 0 | 64 |

Table H‑24 – Values of variable initValue for edge\_dc\_flag ctxIdx

|  |  |  |  |
| --- | --- | --- | --- |
| **Initialization variable** | **edge\_dc\_flag** | | |
| **0** | **1** | **2** |
| **initValue** | 154 | 154 | 154 |

Table H‑25 – Values of variable initValue for edge\_dc\_abs ctxIdx

|  |  |  |  |
| --- | --- | --- | --- |
| **Initialization variable** | **edge\_dc\_abs** | | |
| **0** | **1** | **2** |
| **initValue** | 154 | 154 | 154 |

Table H‑26 – Syntax elements and associated types of binarization, maxBinIdxCtx, ctxIdxTable, and ctxIdxOffset

| **Syntax element** | **initType** | **Type of binarization** | **maxBinIdxCtx** | **ctxIdxTable** | **ctxIdxOffset** |
| --- | --- | --- | --- | --- | --- |
| wedge\_full\_tab\_idx | 0 | FL, cMax = wedgeFullTabIdxBits[ log2PbSize ] (defined in ) | 0 |  | 0 |
| 1 | 0 |  | 1 |
| 2 | 0 |  | 2 |
|  |  |  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |  |  |
| dmm\_dc\_abs | 0 | UEG0 [Ed. (GT) To be specified] | 0 |  | 0 |
| 1 | 0 |  | 1 |
| 2 | 0 |  | 2 |
| dmm\_dc\_sign\_flag | na | FL, cMax = 1 | na | na | na, (Bypass) |
| edge\_code | 0 | TU, cMax = 6 | 0 |  | 0 |
| 1 | 0 |  | 1 |
| 2 | 0 |  | 2 |
| edge\_dc\_flag | 0 | FL, cMax = 1 |  |  | 0 |
| 1 |  |  | 1 |
| 2 |  |  | 2 |
| edge\_dc\_abs | 0 | UEG0 [Ed. (GT) To be specified] | 0 |  | 0 |
| 1 | 0 |  | 1 |
| 2 | 0 |  | 2 |
| edge\_dc\_sign\_flag | na | FL, cMax = 1 | na | na | na, (Bypass) |
| depth\_intra\_mode | 0 | ( specified in subclause ) | 0 |  | 0 |
| 1 | 2 |  | 8 |
| 2 | 2 |  | 16 |
| dmm\_dc\_flag | 0 | FL, cMax = 1 | 0 |  | 0 |
| 1 | 0 |  | 1 |
| 2 | 0 |  | 2 |

Table H‑27 –Values of wedgeFullTabIdxBits[ log2PUSize ]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Initialization variable** | wedgeFullTabIdxBits | | | | |
| **log2PbSize** | 2 | 3 | 4 | 5 | 6 |
| **Value** | 7 | 10 | 11 | 11 | 13 |



Table ‑29 –Binarization for depth\_intra\_mode

|  |  |  |  |
| --- | --- | --- | --- |
| **Name of depth\_intra\_mode** | **Bin String** | | |
| ( cLog2CbSize = =  3 &&  PartMode[ xC ][ yC ] = = PART\_2Nx2N ) | |   ( cLog2CbSize > 3 && cLog2CbSize < 6 ) | cLog2CbSize = = 3 &&  PartMode[ xC ][ yC ] = = PART\_NxN | cLog2CbSize = = 6 |
| INTRA\_DEP\_SDC\_PLANAR | 00 | - | 0 |
| INTRA\_DEP\_NONE | 010 | 0 | 10 |
| INTRA\_DEP\_SDC\_DMM\_WFULL | 011 | - | - |
| INTRA\_DEP\_DMM\_WFULL | 100 | 10 | - |
|  |  |  |  |
|  |  |  |  |
| INTRA\_DEP\_SDC\_DC | 101 | - | 11 |
|  |  |  |  |
| INTRA\_DEP\_CHAIN | 110 | 11 | - |

Table H‑30 –Assignment of ctxIdxInc to syntax elements with context coded bins

| **Syntax element** | **binIdx** | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **>=5** |
| wedge\_full\_tab\_idx | 0 | 0 | 0 | 0 | 0 | 0 |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| dmm\_dc\_flag | 0 | na | na | na | na | na |
| dmm\_dc\_abs | 0 | 0 | 0 | 0 | 0 | 0 |
| dmm\_dc\_sign\_flag | bypass | 0 | 0 | 0 | 0 | 0 |
| edge\_dc\_flag | 0 | na | na | na | na | na |
| edge\_dc\_abs | 0 | 0 | 0 | 0 | 0 | 0 |
| edge\_dc\_sign\_flag | bypass | 0 | 0 | 0 | 0 | 0 |
| edge\_code | 0 | 0 | 0 | 0 | 0 | 0 |