H.7.3.8.5.1 Depth mode parameter syntax

|  |  |
| --- | --- |
| depth\_mode\_parameters( x0 , y0 , log2CbSize ) { | **Descriptor** |
| if( ( log2CbSize = = 3 && PartMode[ xC ][ yC ] = = PART\_2Nx2N )  | | ( log2CbSize > 3 && log2CbSize < 6 ) ) |  |
| **depth\_intra\_mode\_set\_indication\_flag** | ae(v) |
| **depth\_intra\_mode**[ x0 ][ y0 ] | ae(v) |
| if ( DepthIntraMode[ x0 ][ y0 ] = = INTRA\_DEP\_DMM\_WFULL  | |    DepthIntraMode[ x0 ][ y0 ] = = INTRA\_DEP\_SDC\_DMM\_WFULL ) |  |
| **wedge\_full\_tab\_idx**[ x0  ][ y0 ] | ae(v) |
|  |  |
|  | ae(v) |
| else if( DepthIntraMode[ 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( DepthIntraMode[ 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 =   ( DepthIntraMode[ 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.7.4.9.5.1 Depth mode parameter semantics

The variable Log2MaxDmmCbSize is set equal to 5.

The variables depthIntraModeSet is derived as specified in the following:

* If log2CbSize is equal to 6, depthIntraModeSet is set equal to 0.
* Otherwise, if log2CbSize is equal to 3 and PartMode[ xC ][ yC ] is equal to PART\_NxN, depthIntraModeSet is set equal to 1.
* Otherwise depthIntraModeSet is set equal to 2.

**depth\_intra\_mode**[ x0 ][ y0 ] specifies the depth intra mode of the current prediction unit. 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:

* 1. SdcFlag[ x0 ][ y0 ] = ( DepthIntraMode[ x0 ][ y0 ]  = =  INTRA\_DEP\_SDC\_PLANAR )  | |   (‑22)  
      ( DepthIntraMode[ x0 ][ y0 ]  = =  INTRA\_DEP\_SDC\_DMM\_WFULL )

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

* 1. DmmFlag[ x0 ][ y0 ] = ( DepthIntraMode[ x0 ][ y0 ]  = =  INTRA\_DEP\_DMM\_WFULL )  | |   (H‑23)  
      ( DepthIntraMode[ x0 ][ y0 ]  = =  INTRA\_DEP\_DMM\_CPREDTEX )  | |

Table ‑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 |  |
|  |  |  |  |  |  |
| 5 | INTRA\_DEP\_CHAIN | - | 2 | 5 |  |

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

**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 Table H‑3. 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‑24)

**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‑25)

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.

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 name** |
| 0 | INTRA\_PLANAR |
| 1 | INTRA\_DC |
| 2..34 | INTRA\_ANGULAR2..INTRA\_ANGULAR34 |
| 35 | INTRA\_DMM\_WFULL |
|  |  |
| 36 | INTRA\_DMM\_CPREDTEX |
| 37 | INTRA\_CHAIN |

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, 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 if DepthIntraMode[ xPb ][ yPb ] is equal to INTRA\_DEP\_CHAIN, IntraPredModeY[ xPb ][ yPb ] is set equal to INTRA\_CHAIN.
* 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

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

* Otherwise, if predModeIntra is equal to INTRA\_DMM\_WFULL, the corresponding intra prediction mode specified in subclause H.8.4.4.2.7 is invoked with the location ( xTbY, yTbY ), the sample array p and the transform block size nTbS as the inputs and the output are the predicted sample array predSamples.
* Otherwise, if predModeIntra is equal to INTRA\_DMM\_CPREDTEX, the corresponding intra prediction mode specified in subclause H.8.4.4.2.9 is invoked with the location ( xTbY, yTbY ), with the sample array p and the transform block size nTbS as the inputs and the output are the predicted sample array predSamples.
* Otherwise, if predModeIntra is equal to INTRA\_CHAIN, the corresponding intra prediction mode specified in subclause H.8.4.4.2.10 is invoked with the location ( xTbY, yTbY ), with the sample array p and the transform block size nTbS as the inputs and the output are the predicted sample array predSamples.

H.8.4.4.2.7 Specification of intra prediction mode INTRA\_DMM\_WFULL

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,

Output of this process is:

* the predicted samples predSamples[ x ][ y ], with x, y = 0..nTbS − 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 wedgePattern[ x ][ y ] with x, y =0..nTbS − 1, specifying a binary partition pattern is derived as.
     + 1. wedgePattern = WedgePatternTable[ Log2( nTbS ) ][ wedge\_full\_tab\_idx[ xTb ][ yTb ] ] (H‑39)
  2. The depth partition value derivation and assignment process as specified in subclause H.8.4.4.2.11 is invoked with the neighbouring samples p[ x ][ y ], the binary pattern wedgePattern[ xTb ][ yTb ], the transform size nTbS, the dcOffsetAvailFlag set equal to dmm\_dc\_flag[ xTb ][ yTb ], intraChainFlag set equal to 0, and the DC Offsets DcOffset[ xTb ][ yTb ][ 0 ], and DcOffset[ xTb ][ yTb ][ 1 ] as inputs and the output is assigned to predSamples[ x ][ y ].
  3. For x, y = 0..nTbs − 1, inclusive the following applies:
     + WedgeIdx[ xTb + x ][ yTb + y ] is set equal to wedge\_full\_tab\_idx[ xTb ][ yTb ].

H.8.4.4.2.9 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,

Output of this process is:

* the predicted samples predSamples[ x ][ y ], with x, y = 0..nTbS − 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‑43)

* + - 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‑44)

* 1. The depth partition value derivation and assignment process as specified in subclause H.8.4.4.2.11 is invoked with the neighbouring samples p[ x ][ y ], the binary pattern wedgeletPattern[ x ][ y ], the transform size nT, the dcOffsetAvailFlag set equal to dmm\_dc\_flag[ xTb ][ yTb ], intraChainFlag set equal to 0, 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.8.4.4.2.10 Specification of intra prediction mode INTRA\_CHAIN

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,

Output of this process is:

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

The values of the prediction samples predSamples[ x ][ y ], with x, y = 0..nTbS − 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[ xTb ][ yTb ] and edge\_start\_position[ xTb ][ yTb ].
* If edge\_start\_left\_flag[ xTb ][ yTb ] is equal to 1, the start point is set as s[ 0 ][ yS ] where yS is equal to edge\_start\_position[ xTb ][ yTb ].
* Otherwise, the start point is set as s[ xS ][ 0 ] where xS is equal to edge\_start\_position[ xTb ][ yTb ].
  1. The number of edges and the direction of each edge are derived from num\_edge\_codes\_minus1[ xTb ][ yTb ] and edge\_code[ xTb ][ yTb ][ k ], for k = 0.. num\_edge\_codes\_minus1[ xTb ][ yTb ]. The direction of the edge is derived from edge\_code as shown in Table H‑3.
  2. The region boundary generated by connecting each edge separates the predicted samples predSamples[ x ][ y ], with x, y = 0..nTbS − 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..nTbS − 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.8.4.4.2.11 is invoked with the neighbouring samples p[ x ][ y ], the binary pattern edgePattern[ x ][ y ], the transform size nTbS, the dcOffsetAvailFlag set equal to edge\_dc\_flag[ xTb ][ yTb ], intraChainFlag set equal to 1, 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.8.4.4.2.12 Specification of tables WedgePatternTable

NOTE   − Tables and values resulting from the processes specified in the following are independent of any information contained in the bitstream. Therefore the derivation process described in this subclause can be carried out once as part of the initialization of the decoding process. Alternatively, the tables and values can be stored within the decoder (read-only) memory as fixed lookup tables, such that the derivation process described in this section does not need to be implemented in the decoder at all.

The list WedgePatternTable[ log2BlkSize ] of binary partition patterns of size (1  <<  log2BlkSize )x(1  <<  log2BlkSize), the variable NumWedgePattern[ log2BlkSize ] specifying the number of binary partition patterns in list WedgePatternTable[ log2BlkSize ] are derived as specified in the following:

* For log2BlkSize ranging from 2 to Log2MaxDmmCbSize, inclusive, the following applies:
  + Depending on log2BlkSize, the variable resShift is derived as specified in Table H−5.

Table H−5 – Specification of resShift

|  |  |
| --- | --- |
| **log2BlkSize** | **resShift** |
| 2,3 | 1 |
| 4 | 0 |
| Otherwise (5… Log2MaxDmmCbSize) | −1 |

* + The variable wBlkSize is set equal to ( 1  <<  ( log2BlkSize + resShift ) )
  + For wedgeOri in the range of 0 to 5, inclusive, the following ordered steps apply.
    - Depending on wedgeOri the variables xPosS, yPosS, xPosE, yPosE, xIncS, yIncS, xIncE, yIncE are derived as specified in Table H−6.

Table H−6 – Specification of xPosS, yPosS, xPosE, yPosE, xIncS, yIncS, xIncE, yIncE

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **wedgeOri** | 0 | 1 | 2 | 3 | 4 | 5 |
| **xPosS** | 0 | wBlkSize − 1 | wBlkSize − 1 | 0 | 0 | wBlkSize − 1 |
| **yPosS** | 0 | 0 | wBlkSize − 1 | wBlkSize − 1 | 0 | 0 |
| **xPosE** | 0 | wBlkSize − 1 | wBlkSize− 1 | 0 | 0 | 0 |
| **yPosE** | 0 | 0 | wBlkSize − 1 | wBlkSize − 1 | wBlkSize − 1 | 0 |
| **xIncS** | 1 | 0 | −1 | 0 | 1 | 0 |
| **yIncS** | 0 | 1 | 0 | −1 | 0 | 1 |
| **xIncE** | 0 | −1 | 0 | 1 | 1 | 0 |
| **yIncE** | 1 | 0 | −1 | 0 | 0 | 1 |

* + - For m in the range of 0 to wBlkSize − 1, inclusive, the following applies:
      * For n in the range of 0 to wBlkSize − 1, inclusive, the following applies:
        + The Wedgelet pattern generation process as specified in subclause H.8.4.4.2.12.1 is invoked with patternSize being equal to ( 1  <<  log2BlkSize ), the variable resShift, variable wedgeOri, xS being equal to ( xPosS + m \* xIncS ), yS being equal to ( yPosS + m \* yIncS ), xE being equal to ( xPosE + n \* xIncE ) and yE being equal to ( yPosE + n \* yIncE ) as inputs and the output is the binary array curWedgePattern.
        + The wedgelet pattern list insertion process as specified in subclause H.8.4.4.2.12.2 is invoked with log2BlkSize, and the binary partition pattern curWedgePattern as inputs.

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Syntax structure** | **Syntax element** | **ctxTable** | **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 |
| iv\_res\_pred\_weight\_idx |  |  | 0..3 | 4..7 |
| ic\_flag |  |  | 0 | 1 |
| edge\_dc\_flag |  | 0 | 1 | 2 |
| edge\_dc\_abs |  | 0 | 1 | 2 |
| sdc\_residual\_flag |  | 0 | 1 | 2 |
| sdc\_residual\_abs\_minus1 | prefix:  suffix: na | prefix:0 suffix: na | prefix:1 suffix: na | prefix:0 suffix: na |
| inter\_sdc\_flag |  | 0 | 1 | 2 |
| inter\_sdc\_resi\_abs\_minus1 |  | 0 | 1 | 2 |
| inter\_sdc\_resi\_sign\_flag |  | 0 | 1 | 2 |

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

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



Table H‑15 – Values of initValue for dmm\_dc\_abs ctxIdx

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

Table H‑16 – Values of initValue for edge\_code ctxIdx

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

Table H‑22 – Values of initValue for dmm\_dc\_flag ctxIdx

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

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

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

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

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

Table H‑28 – Syntax elements and associated binarizations

| **Syntax structure** | **Syntax element** | **Binarization** | |
| --- | --- | --- | --- |
| **Process** | **Input parameters** |
| coding\_unit( ) | iv\_res\_pred\_weight\_idx | TR | cMax = 2, cRiceParam = 0 |
| ic\_flag | FL | cMax = 1 |
| iv\_res\_pred\_weight\_idx | TR | cMax = 2, cRiceParam = 0 |
| inter\_sdc\_flag | FL | cMax = 1 |
| inter\_sdc\_resi\_abs\_minus1 | UEG0 | [Ed. (GT) To be specified] |
| inter\_sdc\_resi\_sign\_flag | FL | cMax = 1 |
| depth\_mode\_parameters( ) | depth\_intra\_mode\_set\_indication\_flag | FL | cMax = 1 |
| depth\_intra\_mode | TR | cMax = depthIntraModeMaxLen, cRiceParam = 0 |
| wedge\_full\_tab\_idx | FL | cMax = wedgeFullTabIdxBits[ log2PbSize ]  (defined in ) |
|  |  |  |
| edge\_start\_left\_flag [Ed. (GT) Missing! ] |  |  |
| edge\_start\_position [Ed. (GT) Missing! ] |  |  |
| num\_edge\_codes\_minus1 [Ed. (GT) Missing! ] |  |  |
| edge\_code | TR | cMax = 6, cRiceParam = 0 |
| dmm\_dc\_flag | FL | cMax = 1 |
| dmm\_dc\_abs | UEG0 | [Ed. (GT) To be specified] |
| dmm\_dc\_sign\_flag | FL | cMax = 1 |
| edge\_dc\_flag | FL | cMax = 1 |
| edge\_dc\_abs | FL | cMax = 1 |
| edge\_dc\_sign\_flag | FL | cMax = 1 |
| sdc\_residual\_flag | FL | cMax = 1 |
| sdc\_residual\_sign\_flag | FL | cMax = 1 |
| sdc\_residual\_abs\_minus1 | H.9.3.3.10 |  |

Table H‑29 –Values of wedgeFullTabIdxBits[ log2PUSize ]

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



Table H‑31 –Assignment of ctxInc 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 |
| res\_pred\_flag | 0 | na | na | na | na | na |
| ic\_flag | 0 | na | na | na | na | na |
| sdc\_residual\_flag | bypass | na | na | na | na | na |
| sdc\_residual\_abs\_minus1 | prefix: 0 suffix: bypass | prefix: 0 suffix: bypass | prefix: 0 suffix: bypass | prefix: 0 suffix: bypass | prefix: 0 suffix: bypass | prefix: 0 suffix: bypass |
| inter\_sdc\_flag | 0 | 0 | 0 | 0 | 0 | 0 |
| inter\_sdc\_resi\_abs\_minus1 | 0 | 0 | 0 | 0 | 0 | 0 |
| inter\_sdc\_resi\_sign\_flag | 0 | 0 | 0 | 0 | 0 | 0 |