**Draft Text Specification**

The proposed text changes are based on the document JCTVC-R1005-v3.doc for the PU level IntraBC signalling in CE2 Test 2. The changes are marked in yellow.

**Syntax**

#### Coding unit syntax

|  |  |
| --- | --- |
| coding\_unit( x0, y0, log2CbSize ) { | Descriptor |
| if( transquant\_bypass\_enabled\_flag ) |  |
| **cu\_transquant\_bypass\_flag** | ae(v) |
| if( slice\_type != I | | intra\_block\_copy\_enabled\_flag)) |  |
| **cu\_skip\_flag**[ x0 ][ y0 ] | ae(v) |
| nCbS = ( 1  <<  log2CbSize ) |  |
| if( cu\_skip\_flag[ x0 ][ y0 ] ) |  |
| prediction\_unit( x0, y0, nCbS, nCbS ) |  |
| else { |  |
| ~~if( intra\_block\_copy\_enabled\_flag )~~ |  |
| **~~intra\_bc\_flag~~**~~[ x0 ][ y0 ]~~ | ~~ae(v)~~ |
| if( slice\_type != I ~~&& !intra\_bc\_flag[ x0 ][ y0 ]~~ | | intra\_block\_copy\_enabled\_flag) |  |
| **pred\_mode\_flag** | ae(v) |
| if( palette\_enabled\_flag && ChromaArrayType = = 3 &&   CuPredMode[ x0 ][ y0 ] = = MODE\_INTRA ~~&& !intra\_bc\_flag[ x0 ][ y0 ]~~) |  |
| **palette\_mode\_flag**[ x0 ][ y0 ] | ae(v) |
| if( palette\_mode\_flag[ x0 ][ y0 ] ) |  |
| palette\_coding( x0, y0, nCbS ) |  |
| else { |  |
| if( CuPredMode[ x0 ][ y0 ] != MODE\_INTRA | | ~~intra\_bc\_flag[ x0 ][ y0 ] | |~~  log2CbSize = = MinCbLog2SizeY ) |  |
| **part\_mode** | ae(v) |
| if( CuPredMode[ x0 ][ y0 ] = = MODE\_INTRA ~~&& !intra\_bc\_flag[ x0 ][ y0 ]~~) { |  |
| if( PartMode = = PART\_2Nx2N && pcm\_enabled\_flag &&   log2CbSize >= Log2MinIpcmCbSizeY &&  log2CbSize <= Log2MaxIpcmCbSizeY ) |  |
| … |  |

#### 7.3.8.6 Prediction unit syntax

|  |  |
| --- | --- |
| prediction\_unit( x0, y0, nPbW, nPbH ) { | Descriptor |
| if( cu\_skip\_flag[ x0 ][ y0 ] ) { |  |
| if( MaxNumMergeCand > 1 ) |  |
| **merge\_idx**[ x0 ][ y0 ] | ae(v) |
| ~~} else if( intra\_bc\_flag[ x0 ][ y0 ] ) { /\* Intra BC\*/~~ |  |
| ~~bvd\_coding( x0, y0, 2 )~~ |  |
| **~~bvp\_flag~~**~~[ x0 ][ y0 ]~~ | ~~ae(v)~~ |
| } else { /\* MODE\_INTER \*/ |  |
| **merge\_flag**[ x0 ][ y0 ] | ae(v) |
| if( merge\_flag[ x0 ][ y0 ] ) { |  |
| if( MaxNumMergeCand > 1 ) |  |
| **merge\_idx**[ x0 ][ y0 ] | ae(v) |
| } else { |  |
| if( intra\_block\_copy\_enabled\_flag && slice\_type != I && ( PartMode < PART\_NxN | | MinCbLog2SizeY > 3 && PartMode = = PART\_NxN ))) |  |
| **intra\_bc\_flag**[ x0 ][ y0 ] | ae(v) |
| if(intra\_bc\_flag[ x0 ][ y0 ]){ /\* Intra BC\*/ |  |
| bvd\_coding( x0, y0, 2 ) |  |
| **bvp\_flag**[ x0 ][ y0 ] | ae(v) |
| }else{ |  |
| if( slice\_type = = B ) |  |
| **inter\_pred\_idc**[ x0 ][ y0 ] | ae(v) |
| if( inter\_pred\_idc[ x0 ][ y0 ] != PRED\_L1 ) { |  |
| if( num\_ref\_idx\_l0\_active\_minus1 > 0 ) |  |
| **ref\_idx\_l0**[ x0 ][ y0 ] | ae(v) |
| mvd\_coding( x0, y0, 0 ) |  |
| **mvp\_l0\_flag**[ x0 ][ y0 ] | ae(v) |
| } |  |
| if( inter\_pred\_idc[ x0 ][ y0 ] != PRED\_L0 ) { |  |
| if( num\_ref\_idx\_l1\_active\_minus1 > 0 ) |  |
| **ref\_idx\_l1**[ x0 ][ y0 ] | ae(v) |
| if( mvd\_l1\_zero\_flag &&   inter\_pred\_idc[ x0 ][ y0 ] = = PRED\_BI ) { |  |
| MvdL1[ x0 ][ y0 ][ 0 ] = 0 |  |
| MvdL1[ x0 ][ y0 ][ 1 ] = 0 |  |
| } else |  |
| mvd\_coding( x0, y0, 1 ) |  |
| **mvp\_l1\_flag**[ x0 ][ y0 ] | ae(v) |
| } |  |
| } |  |
| } |  |
| } |  |
| } |  |

**Semantics**

#### Coding unit semantics

**~~intra\_bc\_flag~~**~~[ x0 ][ y0 ] equal to 1 specifies that the current coding unit is coded in intra block copying mode. intra\_bc\_flag[ x0 ][ y0 ] equal to 0 specifies that the current coding unit is coded according to pred\_mode\_flag. When not present, the value of intra\_bc\_flag is inferred to be equal to 0. The array indices x0 and y0 specify the location ( x0, y0 ) of the top-left luma sample of the considered coding block relative to the top-left luma sample of the picture.~~

**pred\_mode\_flag** equal to 0 specifies that the current coding unit is coded in inter prediction mode. pred\_mode\_flag equal to 1 specifies that the current coding unit is coded in intra prediction mode. The variable CuPredMode[ x ][ y ] is derived as follows for x = x0..x0 + nCbS − 1 and y = y0..y0 + nCbS − 1:

* ~~If intra\_bc\_flag[ x0 ][ y0 ] is equal to 1, CuPredMode[ x ][ y ] is inferred to be equal to MODE\_INTRA.~~
* ~~Otherwise, i~~If ~~intra\_bc\_flag[ x0 ][ y0 ] is equal to 0 and~~ pred\_mode\_flag is equal to 0, CuPredMode[ x ][ y ] is set equal to MODE\_INTER.
* Otherwise (~~intra\_bc\_flag[ x0 ][ y0 ] is equal to 0 and~~ pred\_mode\_flag is equal to 1), CuPredMode[ x ][ y ] is set equal to MODE\_INTRA.

**part\_mode** specifies partitioning mode of the current coding unit. The semantics of part\_mode depend on CuPredMode[ x0 ][ y0 ]. The variables PartMode and IntraSplitFlag are derived from the value of part\_mode as defined in .

The value of part\_mode is restricted as follows:

* If CuPredMode[ x0 ][ y0 ] is equal to MODE\_INTRA, ~~the following applies:~~
  + ~~If intra\_bc\_flag[ x0 ][ y0 ] is equal to 1, part\_mode shall be in the range of 0 to 3, inclusive.~~
* ~~Otherwise (intra\_bc\_flag[ x0 ][ y0] is equal to 0),~~ part\_mode shall be equal to 0 or 1.
* Otherwise (CuPredMode[ x0 ][ y0 ] is equal to MODE\_INTER), the following applies:
* If intra\_bc\_flag[ x0 ][ y0 ] is equal to 1, part\_mode shall be in the range of 0 to 3, inclusive.
* Otherwise, if log2CbSize is greater than MinCbLog2SizeY and amp\_enabled\_flag is equal to 1, part\_mode shall be in the range of 0 to 2, inclusive, or in the range of 4 to 7, inclusive.

……

#### Prediction unit semantics

**intra\_bc\_flag**[ x0 ][ y0 ] equal to 1 specifies that the current prediction unit is coded in intra block copying mode. intra\_bc\_flag[ x0 ][ y0 ] equal to 0 specifies that the current prediction unit is coded according to pred\_mode\_flag. The array indices x0 and y0 specify the location ( x0, y0 ) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture. When not present, the value of intra\_bc\_flag is inferred as follows:

* + if pred\_mode\_flag is equal to 0 and intra\_block\_copy\_enabled\_flag is equal to 1
    - if slice\_type is equal to I, intra\_bc\_flag is set equal to 1;
    - if PartMode is equal to PART\_NxN and MinCbLog2SizeY is equal to 3, intra\_bc\_flag is set equal to 1;
  + otherwise, intra\_bc\_flag is set equal to 0

**Decoding process**

## Decoding process for coding units coded in intra prediction mode

### General decoding process for coding units coded in intra prediction mode

……

– Otherwise (pcm\_flag[ xCb ][ yCb ] is equal to 0, palette\_mode\_flag[ xCb ][ yCb ] is equal to 0 ), if IntraSplitFlag is equal to 0, the following ordered steps apply:

1. The derivation process for the intra prediction mode as specified in subclause 8.4.2 is invoked with the luma location ( xCb, yCb ) as input.
2. ~~When intra\_bc\_flag[ xCb ][ yCb ] is equal to 1, the derivation process for block vector components in intra block copying prediction mode as specified in subclause is invoked with the luma location ( xCb, yCb ) and variable log2CbSize as inputs, and the output being bvIntra.~~
3. If cu\_residual\_act\_flag[ xCb ][ yCb ] is equal to 1, the following applies:

– For the variable cIdx proceeding over the values 0..2, the following ordered steps apply:

* + The variable comp is set equal to (!cIdx ? L : (cIdx = =1 ? Cb : Cr). [Ed. (GJS): Suggest reformulating to avoid this usage formulation and this variable name.]
  + The general decoding process for intra blocks as specified in subclause is invoked with the location ( xCb, yCb ), the variable log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable predModeIntra set equal to IntraPredModeY[ xCb ][ yCb ], ~~the variable predModeIntraBc set equal to intra\_bc\_flag[ xCb ][ yCb ], the variable bvIntra,~~ the variable cIdx, and variable controlParaACT equal to 1 as inputs, and the output is the residual sample array resSamplescomp.

……

* Otherwise (pcm\_flag[ xCb ][ yCb ] is equal to 0, palette\_mode\_flag[ xCb ][ yCb ] is equal to 0 and IntraSplitFlag is equal to 1), for the variable blkIdx proceeding over the values 0..3, the following ordered steps apply:

……

1. If cu\_residual\_act\_flag[ xCb ][ yCb ] is equal to 1, the following applies:

* For the variable cIdx proceeding over the values 0..2, the following ordered steps apply:
  + The variable comp is set equal to (!cIdx ? L : (cIdx = =1 ? Cb : Cr).
  + The general decoding process for intra blocks as specified in subclause is invoked with the luma location ( xPb, yPb ), the variable log2TrafoSize set equal to log2CbSize − 1, the variable trafoDepth set equal to 1, the variable predModeIntra set equal to IntraPredModeY[ xPb ][ yPb ], ~~the variable predModeIntraBc set equal to 0,~~ the variable cIdx, and variable controlParaACT set equal to 1 as inputs, and the output is the residual sample array resSamplescomp.

……

### Derivation process for block vector components in intra block copying prediction mode

Inputs to this process are:

* a luma location ( xCb, yCb ) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
* a luma location ( xPb, yPb ) of the top-left sample of the current luma prediction block relative to the top-left sample of the current picture,
* the luma coding block size block nCbS,
* two variables nPbSw and nPbSh specifying the width and the height of the luma prediction block,a variable log2CbSize specifying the size of the current luma coding block.

Output of this process is ~~the (nCbS)x(nCbS)~~ (nPbSw)x(nPbSh) array bvIntra.

~~The variables nCbS, nPbSw, and nPbSh are derived as follows:~~

~~nCbS = 1  <<  log2CbSize~~ (8‑25)

~~nPbSw = nCbS / ( PartMode = = PART\_2Nx2N | | PartMode = = PART\_2NxN ? 1 : 2 ) (8‑25)~~

~~nPbSh = nCbS / ( PartMode = = PART\_2Nx2N | | PartMode = = PART\_Nx2N ? 1 : 2 ) (8‑25)~~

The variable BvpIntra[ compIdx ] specifies a block vector predictor. The horizontal block vector component is assigned compIdx = 0 and the vertical block vector component is assigned compIdx = 1.

~~Depending upon PartMode, the variable numPartitions is derived as follows:~~

~~– If PartMode is equal to PART\_2Nx2N, numPartitions is set equal to 1.~~

~~– Otherwise, if PartMode is equal to either PART\_2NxN or PART\_Nx2N, numPartitions is set equal to 2.~~

~~– Otherwise (PartMode is equal to PART\_NxN), numPartitions is set equal to 4.~~

~~The array of block vectors bvIntra is derived by the following ordered steps, for the variable blkIdx proceeding over the values 0..( numPartitions − 1 ):~~

* ~~The variable blkInc is set equal to ( PartMode = = PART\_2NxN ? 2 : 1 ).~~
* ~~The variable xPb is set equal to xCb + nPbSw \* ( blkIdx \* blkInc % 2 ).~~
* ~~The variable yPb is set equal to yCb + nPbSh \* ( blkIdx / 2 )~~

1. The following ordered steps apply, for the variable compIdx proceeding over the values 0..1:
2. The variable LastBvIntra[ 0 ][ compIdx ] and LastBvIntra[ 1 ][ compIdx ]specifies the last two block vector predictor. If this process is invoked for the first time for the current coding tree unit, LastBvIntra[ compIdx ] is derived as follows:

LastBvIntra[ 0 ][ 0 ] = −2\* nCbS; LastBvIntra[ 0 ][ 1 ] = 0

LastBvIntra[ 1 ][ 0 ] = −nCbS; LastBvIntra[ 1 ][ 1 ] = 0

~~Depending upon the number of times this process has been invoked for the current coding tree unit,~~ subclause 8.4.4.1 is invoked with the luma coding block location ( xCb, yCb ), the coding block size nCbS, the luma prediction block location ( xPb, yPb ), the luma prediction block width nPbSw, the luma prediction block height nPbSh, the last block vectors LastBvIntra, and the partition index blkIdx as inputs, and the block vector predictor BvpIntra[ xPb ][ yPb ] as the output, and bvIntra[ xPb ][ yPb ][ compIdx ] is set equal to BvdIntra[ xPb ][ yPb ][ compIdx ] + BvpIntra[ xPb ][ yPb ][ compIdx ]

……

### Decoding process for intra blocks

#### General decoding process for intra blocks

Inputs to this process are:

– a sample location ( xTb0, yTb0 ) specifying the top-left sample of the current transform block relative to the top‑left sample of the current picture,

– a variable log2TrafoSize specifying the size of the current transform block,

– a variable trafoDepth specifying the hierarchy depth of the current block relative to the coding unit,

– a variable predModeIntra specifying the intra prediction mode,

– ~~a variable predModeIntraBc specifying the intra block copying mode,~~

~~– a variable bvIntra specifying the intra block copying vector,~~

– a variable cIdx specifying the colour component of the current block, and

……

1. When controlParaACT is not equal to 1, depending upon the value of predModeIntraBc, the following applies:

– ~~When predModeIntraBc is equal to 0,~~ the general intra sample prediction process as specified in subclause 8.4.4.2.1 is invoked with the transform block location ( xTb0, yTb0 + yTbOffset ), the intra prediction mode predModeIntra, the transform block size nTbS, and the variable cIdx as inputs, and the output is an (nTbS)x(nTbS) array predSamples.

– ~~Otherwise (predModeIntraBc is equal to 1), the intra block copying process as specified in subclause  is invoked with the transform block location ( xTb0, yTb0 + yTbOffset ), the transform block size nTbS, the variable trafoDepth, the variable bvIntra, and the variable cIdx as inputs, and the output is an (nTbS)x(nTbS) array predSamples.~~

……

### Derivation process for motion vector components and reference indices

#### 8.5.3.2.1 General

……

For the derivation of the variables mvL0 and mvL1, refIdxL0 and refIdxL1, as well as predFlagL0 and predFlagL1, the following applies:

* If merge\_flag[ xPb ][ yPb ] is equal to 1, the derivation process for luma motion vectors for merge mode as specified in subclause is invoked with the luma location ( xCb, yCb ), the luma location ( xPb, yPb ), the variables nCbS, nPbW, nPbH, and the partition index partIdx as inputs, and the output being the luma motion vectors mvL0, mvL1, the reference indices refIdxL0, refIdxL1, and the prediction list utilization flags predFlagL0 and predFlagL1.
* Otherwise, if intra\_bc\_flag[ xPb + x ][ yPb + y ] is equal to 1, The derivation process for block vector components as specified in subclause 8.4.4 is invoked with the luma coding block location ( xCb, yCb ), the luma prediction block location ( xPb, yPb ), the luma coding block size block nCbS, the luma prediction block width nPbW and the luma prediction block height nPbH as inputs, and the block vector array bvIntra as output.
* Otherwise, for X being replaced by either 0 or 1 in the variables predFlagLX, mvLX, and refIdxLX, in PRED\_LX, and in the syntax elements ref\_idx\_lX and MvdLX, the following applies:

……

When ChromaArrayType is not equal to 0 and predFlagLX, with X being 0 or 1, is equal to 1, the derivation process for chroma motion vectors in subclause is invoked with mvLX as input, and the output being mvCLX.

If merge\_flag[ xPb ][ yPb ] is equal to 1 and the variable intra\_bc\_flag for the chosen merge candidate is equal to 1, intraBCFlag is set equal to 1; otherwise, intraBCFlag is set equal to 0. the following assignments are made for x = 0..nPbW − 1 and y = 0..nPbH − 1:

intra\_bc\_flag[ xPb + x ][ yPb + y ] = intraBCFlag (8‑xx)

If intraBCFlag is equal to 1,

bvIntra[ xPb + x ][ yPb + y ] = mvL0 (8‑xx)

#### Decoding process for inter prediction samples

##### General

……

Let predSamplesL0L and predSamplesL1L be (nPbW)x(nPbH) arrays of predicted luma sample values and when ChromaArrayType is not equal to 0, predSamplesL0Cb, predSamplesL1Cb, predSamplesL0Cr, and predSamplesL1Cr be (nPbW / SubWidthC)x(nPbH / SubHeightC) arrays of predicted chroma sample values.

* If intra\_bc\_flag [ xCb + xBl ][ yCb + yBl ] is equal to 1, the following applies:
  + The array prediction sample array for each colour component is derived by invoking the subclause with the luma locations ( xCb + xBl, yCb + yBl ), the luma prediction block width nPbW, the luma prediction block height nPbH, the motion vectors bvIntra, and the current reconstructed picture array currPicL, the colour component index of the current block cIdx as inputs.
* Otherwise, for X being each of 0 and 1, when predFlagLX is equal to 1, the following applies:

……

**Parsing process**

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** |
| sao( ) | sao\_merge\_left\_flag sao\_merge\_up\_flag |  | 0 | 1 | 2 |
| sao\_type\_idx\_luma sao\_type\_idx\_chroma |  | 0 | 1 | 2 |
| coding\_quadtree( ) | split\_cu\_flag[ ][ ] |  | 0..2 | 3..5 | 6..8 |
| coding\_unit( ) | cu\_transquant\_bypass\_flag |  | 0 | 1 | 2 |
| cu\_skip\_flag |  |  | 0..2 | 3..5 |
| ~~intra\_bc\_flag[ ][ ]~~ |  | ~~0~~ | ~~1~~ | ~~2~~ |
| palette\_mode\_flag[ ][ ] |  | 0 | 1 | 2 |
| pred\_mode\_flag |  |  | 0 | 1 |
| part\_mode |  | 0 | 1..4 | 5..8 |
| prev\_intra\_luma\_pred\_flag[ ][ ] |  | 0 | 1 | 2 |
| intra\_chroma\_pred\_mode[ ][ ] |  | 0 | 1 | 2 |
| rqt\_root\_cbf |  |  | 0 | 1 |
| cu\_residual\_act\_flag |  | 0 | 1 | 2 |
| prediction\_unit( ) | merge\_flag[ ][ ] |  |  | 0 | 1 |
| merge\_idx[ ][ ] |  |  | 0 | 1 |
| intra\_bc\_flag[ ][ ] |  |  | 1 | 2 |
| inter\_pred\_idc[ ][ ] |  |  | 0..4 | 5..9 |
| ref\_idx\_l0[ ][ ], ref\_idx\_l1[ ][ ] |  |  | 0..1 | 2..3 |
| mvp\_l0\_flag[ ][ ], mvp\_l1\_flag[ ][ ], bvp\_flag[ ][ ] |  |  | 0 | 1 |

| Table 9‑38 – Syntax elements and associated binarizations | | | |
| --- | --- | --- | --- |
| **Syntax structure** | **Syntax element** | **Binarization** | |
| **Process** | **Input parameters** |
| coding\_quadtree( ) | split\_cu\_flag[ ][ ] | FL | cMax = 1 |
| coding\_unit( ) | cu\_transquant\_bypass\_flag | FL | cMax = 1 |
| cu\_skip\_flag | FL | cMax = 1 |
| ~~intra\_bc\_flag~~ | ~~FL~~ | ~~cMax = 1~~ |
| palette\_mode\_flag | FL | cMax = 1 |
| …… |  |  |
| palette\_coding( ) | previous\_palette\_entry\_flag[] | FL | cMax = 1 |
| …… |  |  |
| prediction\_unit( ) | merge\_flag[ ][ ] | FL | cMax = 1 |
| merge\_idx[ ][ ] | TR | cMax = MaxNumMergeCand − 1, cRiceParam = 0 |
| intra\_bc\_flag | FL | cMax = 1 |
| inter\_pred\_idc[ x0 ][ y0 ] |  | nPbW, nPbH |
| ref\_idx\_l0[ ][ ] | TR | cMax = num\_ref\_idx\_l0\_active\_minus1, cRiceParam = 0 |
| mvp\_l0\_flag[ ][ ] | FL | cMax = 1 |
| ref\_idx\_l1[ ][ ] | TR | cMax = num\_ref\_idx\_l1\_active\_minus1, cRiceParam = 0 |
| mvp\_l1\_flag[ ][ ] | FL | cMax = 1 |
| bvp\_flag[ ][ ] | FL | cMax = 1 |

Table 9‑38 – Values of initValue for intra\_bc\_flag

|  |  |  |  |
| --- | --- | --- | --- |
| **Initialization variable** | **ctxIdx of intra\_bc\_flag** | | |
| **~~0~~** | **1** | **2** |
| **initValue** | ~~185~~ | 197 | 197 |

Table 9‑40 – Binarization for part\_mode

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **CuPredMode** [ xCb ][ yCb ] | **intra\_bc\_flag** [ xCb ][ yCb ] | **part\_mode** | **PartMode** | **Bin string** | | | |
| log2CbSize >  MinCbLog2SizeY | | log2CbSize  = =  MinCbLog2SizeY | |
| !amp\_enabled\_flag | amp\_enabled\_flag | log2CbSize  = =  3 | log2CbSize > 3 |
| MODE\_INTRA | 0 | 0 | PART\_2Nx2N | - | - | 1 | 1 |
| 1 | PART\_NxN | - | - | 0 | 0 |
| MODE\_INTER | 1 | 0 | PART\_2Nx2N | 1 | 1 | 1 | 1 |
| 1 | PART\_2NxN | 01 | 01 | 01 | 01 |
| 2 | PART\_Nx2N | 00 | 00 | 001 | 001 |
| 3 | PART\_NxN | - | - | 000 | 000 |
| 0 | 0 | PART\_2Nx2N | 1 | 1 | 1 | 1 |
| 1 | PART\_2NxN | 01 | 011 | 01 | 01 |
| 2 | PART\_Nx2N | 00 | 001 | 001 | 001 |
| 3 | PART\_NxN | - | - | - | 000 |
| 4 | PART\_2NxnU | - | 0100 | - | - |
| 5 | PART\_2NxnD | - | 0101 | - | - |
| 6 | PART\_nLx2N | - | 0000 | - | - |
| 7 | PART\_nRx2N | - | 0001 | - | - |