#### General slice segment header syntax

**…..**

|  |  |
| --- | --- |
| **slice\_loop\_filter\_across\_slices\_enabled\_flag** | u(1) |
| } |  |
| if( tiles\_enabled\_flag | | entropy\_coding\_sync\_enabled\_flag ) { |  |
| **num\_entry\_point\_offsets** | ue(v) |
| if( num\_entry\_point\_offsets > 0 ) { |  |
| **offset\_len\_minus1** | ue(v) |
| for( i = 0; i < num\_entry\_point\_offsets; i++ ) |  |
| **entry\_point\_offset\_minus1**[ i ] | u(v) |
| } |  |
| } |  |
| if ( intra\_block\_copy\_enabled\_flag) { |  |
| **mvd\_bvd\_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( ) |  |
| } |  |

#### 7.4.7.1 General slice segment header semantics

**mvd\_bvd\_flag** specifies the motion vector and the block vector entropy residual decoding method.

**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\*/ |  |
| ~~b~~mvd\_coding( x0, y0, 2 ) |  |
| 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( 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) |
| } |  |
| } |  |
| } |  |
| } |  |

**7.3.8.10 Motion vector and block vector difference syntax**

|  |  |
| --- | --- |
| mvd\_coding( x0, y0, refList ) { | **Descriptor** |
| if(mvd\_bvd\_flag){ |  |
| **abs\_bvd\_greater0\_flag**[ 0 ] | ae(v) |
| **abs\_bvd\_greater0\_flag**[ 1 ] | ae(v) |
| if( abs\_bvd\_greater0\_flag[ 0 ] ) { |  |
| **abs\_bvd\_minus1**[ 0 ] | ae(v) |
| **bvd\_sign\_flag**[ 0 ] | ae(v) |
| } |  |
| if( abs\_bvd\_greater0\_flag[ 1 ] ) { |  |
| **abs\_bvd\_minus1**[ 1 ] | ae(v) |
| **bvd\_sign\_flag**[ 1 ] | ae(v) |
| } |  |
| } else { |  |
| **abs\_mvd\_greater0\_flag**[ 0 ] | ae(v) |
| **abs\_mvd\_greater0\_flag**[ 1 ] | ae(v) |
| if( abs\_mvd\_greater0\_flag[ 0 ] ) |  |
| **abs\_mvd\_greater1\_flag**[ 0 ] | ae(v) |
| if( abs\_mvd\_greater0\_flag[ 1 ] ) |  |
| **abs\_mvd\_greater1\_flag**[ 1 ] | ae(v) |
| if( abs\_mvd\_greater0\_flag[ 0 ] ) { |  |
| if( abs\_mvd\_greater1\_flag[ 0 ] ) |  |
| **abs\_mvd\_minus2**[ 0 ] | ae(v) |
| **mvd\_sign\_flag**[ 0 ] | ae(v) |
| } |  |
| if( abs\_mvd\_greater0\_flag[ 1 ] ) { |  |
| if( abs\_mvd\_greater1\_flag[ 1 ] ) |  |
| **abs\_mvd\_minus2**[ 1 ] | ae(v) |
| **mvd\_sign\_flag**[ 1 ] | ae(v) |
| } |  |
| } |  |
| } |  |