Intel® OpenSource HD Graphics Programmer's Reference Manual (PRM)

Volume 2 Part 3: Multi-Format Transcoder – MFX (Ivy Bridge)

For the 2012 Intel® Core™ Processor Family

May 2012

Revision 1.1

NOTICE:
This document contains information on products in the design phase of development, and Intel reserves the right to add or remove product features at any time, with or without changes to this open source documentation.

Doc Ref #: IHD-OS-V2 Pt 3_r1.1 – 05 12

Creative Commons License

You are free to Share — to copy, distribute, display, and perform the work

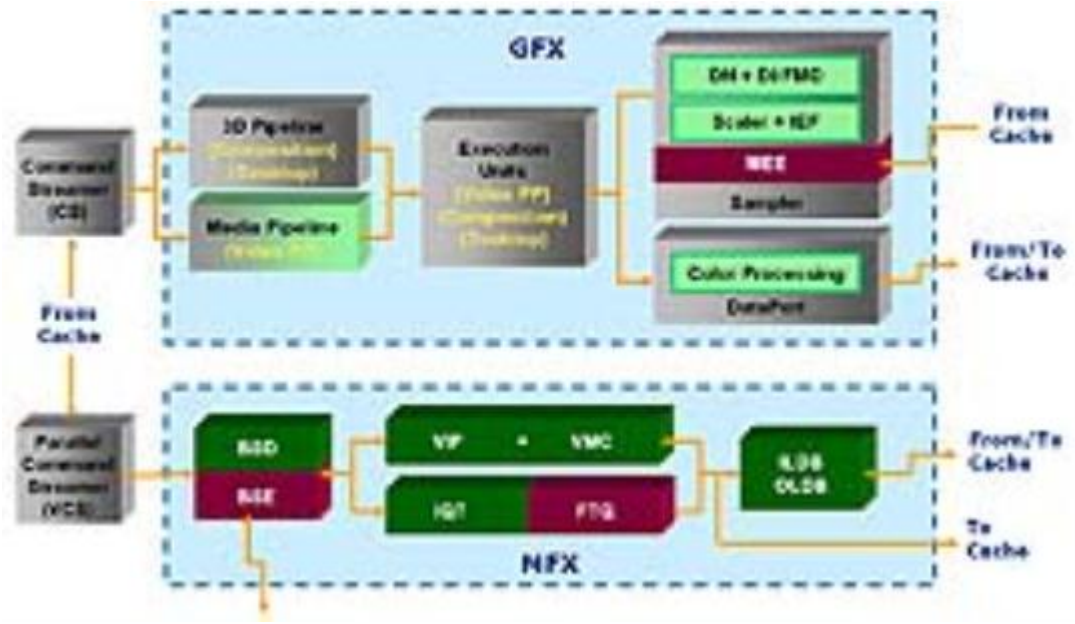
Under the following conditions:

Attribution. You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

No Derivative Works. You may not alter, transform, or build upon this work.

Contents

1. MFX Introduction



Multi-Format Codec (MFX) Engine is the hardware fixed function pipeline for decode and encoding. It includes multi-format decoding (MFD) and multi-format encoding (MFC).

1.1 MFD Overview

When used for decoding, we refer to the MFX Engine also as the MFD Engine.

The Multi-Format Decoder (MFD) is a hardware fixed function pipeline for decoding the three video codec formats and one image compression codec format : AVC (H.264), VC-1, MPEG2 and JPEG.

- Compliant to next generation high definition optical video disc requirements (e.g.) with sufficient performance headroom
    - Support AVC 4:2:0 Main and High (8-bit only) Profile only (no support for Baseline, Extended and High-10 Profiles), up to Level 5.1 (max 983,040 MB/s, max 36,864 MB/frame, and at most one dimension can reach 4K pixel) resolution and up to 40 mbps bitstream. With sufficient duty cycles, higher bit rate contents can also be decoded.
        - Allow a B-picture (frame or field) as a reference picture
        - MVC is supported by AVC Long Format. All MVC specific functions are taken care of by the Application.
    - Support VC1 4:2:0 Simple, Main and Advanced Profiles, up to Level 4 (max 491,520 MB/s and max 16,384 MB/frame; max only one dimension will be at 4K pixel) resolution and up to 40 mbps bitstream. With sufficient duty cycles, higher bit rate contents can also be decoded.

- Allow a B-field as a reference picture only in interlaced field decoding, no other modes.

o Support MPEG2 HD Main Profile (4:2:0), up to High Level (1920x1152 pixels) and up to 80 mbps bitstream. With sufficient duty cycles, higher bit rate contents can also be decoded. No support for SNR and spatial-scalability.

- Does not support B-picture as a reference picture.

o Support Baseline JPEG with five choma types (4:0:0, 4:1:1, 4:2:2, 4:2:0, and 4:4:4. No support for Extended DCT-based mode, Progressive mode, Loseless mode, nor Hierarchical mode

- H/W support 64Kx64K, but Surface State can support only up to 16kx16k

Features	Supported	Unsupported
Coding processes	Baseline sequential mode: • 8-bit pixel precision of source images • loadable 2 AC and 2 DC Huffman tables • 3 loadable quantization matrix for Y, U, V • Interleaved and non-interleaved Scans • Single and multiple Scans	Extended DCT-based mode, Lossless, Hierarchical modes: More than 8 bit pixel resolution, progressive mode, arithmetic coding, 4 AC and 4 DC Huffman tables (extended mode), predictive process (lossless), multiple frames (hierarchical)
Number of image channels	1 for grey image 3 for Y, Cb, Cr color image	4-th channel (usually alpha blending image)
Image resolution	Arbitrary image size up to 16K * 16K	Larger than 16K * 16K (64K * 64K is max. in the JPEG standard)
Chroma subsampling ratio	Chroma 4:0:0 (grey image) Chroma 4:1:1 Chroma 4:2:0 Chroma horizontal 4:2:2 Chroma vertical 4:2:2 Chroma 4:4:4	Any other arbitrary ratio, e.g., 3:1 subsampled chroma
Additional feature (post-processing)	Image rotation: 90/180/270 degrees	

o H/W does not impose restriction on picture frame aspect ratio, but is bounded by a max 256 MBs (4096 pixels) per dimension programmable at the H/W interface specifications.

- For example, supporting HD video resolution 1920x1080/60i, 1920x1080/24p, 1280x720/60p

- Performance requirements with MFX core frequency above 1GHz

o Real-time performance around 10% duty cycle

o Support concurrently decoding of at least two active HD bitstreams of different formats (For example, one AVC and one VC1 HD bitstream)

- The parsing of transport layer and sequence layer is not performed in this hardware, and is required to be done in the host software. We have added the parsing of Slice Header for AVC and the Picture+Slice Header for VC1.
- The MFD hardware pipeline is operated concurrently with and independently from the Graphics (3D/Media) pipeline with separate command streamer. The two parallel engines are designed with the similar command protocol. They can be executed in parallel with different context.
- Local storages and buffers along the hardware pipeline are kept at minimum. For example, there is no on-die row-store memory. They are resided on the system memory. MFD is designed to hide the memory access latency (in both the row stores and in the motion compensation units) in maximizing its decoding throughput.
- Support the following operating modes
    - VLD mode – operation starts from entropy decoding of the compressed bit stream (parsing Slice Header and Slice Data Layer in AVC, Picture layer, Slice layer and MB Layer in VC-1, and MB-layer in MPEG2), all the way, to the reconstruction of display picture, including in-loop , if any.
        - Streamout mode – a new feature of the VLD mode in assisting transcoding during decoding. Selected uncompressed data (e.g. per MB MV information) will be sent out to the EU and the ME engine (resided on the Sampler of the 3D Gx Pipeline) for encoding into a different format or for the purpose of transcaling and transrating. In addition, the uncompressed result may continue to be processed by the rest of pipeline as in VLD mode to generate the display picture for transcoding. That is, while intermediate data are streaming out to the memory, the MFD Engine continues its decoding as ususal.
        - For JPEG, only VLD mode is supported (No IT mode). Host software decodes Frame and Scan layers (down to Scan header in the JPEG bit stream syntax) and sends all the corresponding information and Scan payload to the MFD hardware pipeline.
    - IT mode – when host software has already performed all the bit stream parsing of the compressed data and packaging the uncompressed result into a specific format (as a sequence of per-MB record) stored in memory. The hardware pipeline will fetch one MB record at a time and perform the rest of the decoding process as in VLD mode
    - Host software (Application) is responsible for parsing and decoding all the transport and program layers, and all sequence layers. These parameters are passed to Driver and forwarded to H/W as needed through different STATE commands. Host software is also responsible for separating non-video data (audio, meta and user data) from sending to H/W.
        - MFD Engine is only responsible for macro-block and block layers decoding, plus certain level of header decoding. For AVC MFD starts decoding from Slice Header; for VC1, MFD starts decoding from Picture Header, and for MPEG2 decoding starts from MB Layer only.
        - For JPEG, MFD is responsible for ECS and block layers decoding.
- Support bitstream formats (compressed video data) for each codec
    - AVC – 2 formats
        - DXVA2 AVC Short Slice Format Specification (new in)
        - DXVA2 AVC Long Slice Format Specification

- MVC – 2 formats
  - DXVA2 AVC Long Slice Format Specification (exactly the same as AVC)
- VC1 – 2 formats
  - DXVA2 VC1 Specification (new in)
    - Fully compliant to Picture Parameter and Slice Control Parameter interface definition
- MPEG2
  - MB Layer only, according to DXVA 1 Specification
- JPEG
  - Intel proprietary format (new in) this generation.
  - ECS Layer

- The MFX codec is designed to be a stateless engine, that it does not retain any history of settings (states) for the encoding/decoding process of a picture. Hence, driver must issue the full set of MFX picture state command sequence prior to process each new picture. In addition, driver must issue the full set of Slice state command sequence prior to process a slice.
  - In particularly, RC6 always happens between frame boundaries. So at the beginning of every frame, all state information needs to be programmed. There is no state information as part of media context definition.
- To activate the AVC deblocker logic for incoming uncompressed 4:2:0-only video stream, one can pack the uncompressed video stream to compliant with the IPCM MB data format (including ILDB control information) and feed them into the MFD engine in IT mode. Since the MFD Engine is in IPCM mode, transformation, inter and intra processing are all inactive.

Start Code Detection and removal are done in the CPU, but the Start Code Emulation Prevention Byte is detected and removed by the front end logic in the MFD. The bitstream format for each codec and for each mode is specified in this document.

Codec specific information are based on the following released documents from third parties :

- Draft of Version 4 of H.264/AVC (ITU-T Recommendation H.264 and ISO/IEC 14496-10 (MPEG-4 part 10) Advanced Video Coding); JVT-O205d1.doc; dated 2005-05-30
- Final Draft SMPTE Standard : VC1 Compressed Video Bitstream Format and Decoding Process, SMPTE 421M, dated 2006-1-6; PDF file.
- MPEG2 Recommendation ITU T H.262 (1995 E), ISO/IEC 13818-2: 1995 (E); doc file.
- Digital Compression and Coding of Continuous-tone Still Images, ITU-T Rec. T.81 and ISO/IEC 10918-1: Requirements and guidelines September 18 1992; itu-t81[1].pdf

1.1.1 MFD Memory Interface

The Memory Arbitrator follows the pre-defined arbitration policy (as indicated in the following listing P0 to P11, in which P0 is the highest priority) to select the next memory request to service, then it will perform the TLB translation (translation to physical address in memory), and make the actual request to memory.

The Memory Arbitration unit is also responsible for capturing the return data from memory (read request) and forward it to the appropriate unit along the MFD Engine.

- Read streams: (all 64B requests)
    - Commands for BSD : linear (including indirect data) (P0)
        - Indirect DMA (P1)
    - Row store for BSD: linear (P5)
    - Row store for MPR: linear (P6)
    - MC ref cache fetch : tiled (P2)
    - Intra row store: linear (P9)
    - ILDB row store: linear (P10)
- Write streams: (all 64B requests)
    - Row store write for BSD: linear and can avoid partial writes (P3)
    - Row store write for MPR: linear and can avoid partial writes (P4)
    - Intra row store write: linear and can avoid partial writes (P7)
    - ILDB row store write: linear and can avoid partial writes (P8)
    - Final dest writes: tiled and can potentially be partial, two ways to avoid these partials: 1) either write garbage and buffers are aligned or 2) read-modify writes for dribble end of line cases (P11)

1.1.2 MFD Codec-Specific Commands

MFD hardware pipeline supports 3 different codec standards : AVC, VC1 and MPEG2. To make the interface flexible, each codec is designed with its own set of commands.

There are two categories of commands for each codec format : one set for VLD mode and one set for IT mode.

1.2 MFC Overview

Multi-Format Codec (MFX) Engine is the hardware fixed function pipeline for decode and encoding. It includes multi-format decoding (MFD) and multi-format encoding (MFC). Many decoding function blocks in MFD such as VIP, VMC, IQT, etc, are also used in encoding mode. Two blocks FTQ and BSE are encoding only.

The encoding process is partitioned across host software, GPE engine and the MFX engine. The generation of transport layer, sequence layer, picture layer and slice header layer is required to be done in the host software. GP hardware is responsible for compressing from Slice Data Layer down to all macro-block and block layers. Specifically, GPE w/ VME acceleration is for motion vector estimation, motion estimation, and code decision. The **VME**(>*Video Motion Estimation*) is located next to all image processing units, such as DN (*denoise*) and DI (*deinterlace*) in sampler in GPE. MFX is for final bit packing and reconstructed picture generation.

MFC is operated concurrently with and independently from the GPE (3D/Media) pipeline with separate command streamer. The two parallel engines have similar command protocol. They can be executed in parallel with different context. For encoding, motion search, MB mode decision and rate control are performed using GPE pipeline resources.

MFC is implemented to achieve the following objectives.

- Compliant to next generation high definition optical video disc requirements (e.g.) with sufficient performance headroom

- Support AVC 4:2:0 Main Profile and High Profile only (8-bit only), up to Level 4.1 resolution and up to 40 mbps bitstream. With sufficient duty cycles, higher bit rate contents can also be encoded. There is no support for Baseline, Extended and High-10 Profiles.

- Performance requirements with MFX core frequency above 667MHz
  - Real-time performance with 20% duty cycle or less
  - Support concurrently decoding of two active HD bitstreams of different formats (For example, one AVC and one VC1 HD bitstream) and one active HD encoding.

As the result of this hardware partitioning, VPP and ENC are always running in GPE, and PAK is what runs exactly in MFC.

PAK – residue packing and entropy coding, including block transformation, quantization, data prediction, bitrate tuning and reference decoding. It delivers final packed bitstream and decoded key-frame reference.

- As the same as ENC, PAK is invoked on a Slice boundary; a single call of VPP can lead to multiple calls for PAK.
- Rate control is inside ENC and PAK only, not in VPP
- PAK must always perform with reconstructed reference picture

There is a general dependency of the three operation pipelines. Semaphores are inserted either according to frames or slices. The main CS will also be notified when the decoded reference is ready for the next frame set to be encoded. The detailed discussion will be found in a later section.

Host software is responsible for encoding the transport stream and all the sequence, picture and slice layer/header in the bit-stream; the MFC system is responsible for compressing from Slice Data Layer down to all macro-block and block layers.

1.2.1 Example Usage Model

Encoding flow described here assumes that input stream is a series of uncompressed video frames that will be converted into YUV (4:2:0) for encoding. Depending upon how this stream is derived, application usage may be listed as below:

- Single video stream encoder, video capture+encode, home movie making (SD/HD)
- PVR usage: Decode the incoming stream to generate YUV (uncompressed) frames and then encode to have a compressed file size storage (also transcoding)
- The HW asset needs to support single stream decode (SD+HD) and independent stream encode (HD). This usage can be enabled by scheduling HW decoder at command stream level instead of HW managed time-slicing.

For illustration purpose only, here are two possible usage modes: *user-friendlymode* and *professional mode*.

- ***Professionalmode* (PFM)**:
  Application does the picture order sequencing and submit the picture frame-by-frame to VPP as IN coded order with specified frame coding type, and it has the full custom control of the GOP structure
  - no restriction on numbers of I, P and B
  - no restriction on individual interlace and progressive picture

- ***User-friendlymode* (UFM)**:
  Application presents video in display order. In this case, the application can only specify two pre-defined parameters: **NumP** and **NumB**, for the underlining pre-defined GOP structure.

    - Where **NumP** is the number of P (or P/P) -frames in a GOP, and **NumB** is the number of B (or B/B) frames between two consecutive key (I, P, I/I, I/P, or P/P) frames.

In this case, the driver will need to composite the final GOP structure based on the application parameters, and need to perform the proper sequencing of picture to the VPP in the coding order (i.e. it will hold the pictures in the memory and submit the correct picture buffer address only in coding order), then pass the data in as the same as in PFM.

A GOP (group of pictures) is a complete encoding unit consisting of a number of video frames. In general a GOP structure has the following form:

I0, B-B1, K1, B-B2, K2, B-B3, K3, … , B-BN, KN

in display order, or equivalently

I¬0, K1, B-B1, K2, B-B2, K3, B-B3, … , KN, B-BN

in coded storage/transmission order. Where K is a key (i.e. I or P) frame, and B-Bi is a set of Mi consecutive B frames. Thus, there are 1+N+(M1+…+MN) frames in a GOP.

In the UFM, we have N = NumP, and Mk = NumB for all k. Where NumB must be an number from 0, 1, 2, or 3. For examples:

- NumP = 5, NumB = 2: GOP = I0 P3 B1 B2 P6 B4 B5 P9 B7 B8 P12 B10 B11 P15 B13 B14 I16 …
- NumP = 7, NumB = 0: GOP = I0 P1 P2 P3 P4 P5 P6 P7 I8 P9 …
- NumP = 0, NumB = 0: GOP = I0 I1 I2 I3 I4 I5 I6 I7 I8 I9 …

As a result, a unified hardware interface is given.

All frame/slice type determination/specifications are performed prior to the hardware interface in coded order.

1.2.2 Sample Algorithmic Flow

Assuming all the hardware components are given, there are infinite usage possibilities left with intention for software to decide according to its own application needs depending upon the balanced requirement of coding speed, frame latency, power-consumption, and video quality, and depending upon the usage modes and user preferences (such as low-frame-rate-high-frame-quality vs. high-frame-rate-low-frame-quality).

The last part of this chapter, we illustrate a generic sample to show how a compression algorithm can be implemented to use our hardware.

Step 1. Application or driver initializes the encoder with desired configuration, including speed, quality, targeted bit-rate, input video info, and output format and restrictions.

Step 2. VPP – Application or diver feeds VPP one frame at a time in coded order with specified frame or field type, as well as transcoding informations: motion vectors, coded complexity (i.e. bit size).
It will perform denoising and deblocking based on original and targeted bit-rate, and output additional 4 spatial variances and 2 temporal variances for each macroblock as well as the whole frame.

Step 3. ENC – Application or diver feeds ENC one coding slice buffer at a time including all VPP output. The frame level data is accessible to all slices.

a. Encoding setup unit (**ESE**) will set picture level quality parameters (including LUTs, and othe costing functions) and set target bit-budget (TBB) and maximal bit-budget (MBB) to each macroblock based on rate-control (**RC**) scheme implemented. For B-frames, it wll also make ME searching mode decision (either Fast, Slow or Uni-directional).

b. Loop over all macroblocks: calculate searching center (**MVP**) perform individual ME and IE (**MEE**). Multi-thread may be designed for HW according to a zigzag order for minimal dependency issue.

c. ENC make microblock level code decision (**CD**) outputs macroblock type, intra-mode, motion-vectors, distortions, as well as TBBs and MBBs.

Step 4. PAK – Application or diver feeds PAK one array of coded macroblocks covering a slice at a time, including all ENC output. Original frame buffer and reconstructed reference frame buffers are also available for PAK to access.

a. PAK may create bitstreams for all sequence, gop, picture, and slice level headers prior the first macroblock.

b. Loop over all macroblocks, accurate prediction block is constructed for either inter- or intra- predictions (**VMC** & **VIP**). If MB distortion is less than some predetermined threshold, for a B slice this step can be skiped as well as the Steps (c)-(e) and jump directly to Step (f); for a key slice the prediction calculated here will be directly used as the reference thus it jump to Step (e) after this step.

c. Differencing the predicted block from the original block derives the residue block. Forward transformation and quantization (**FTQ**) is performed. For B slice, it will jump to Step (f) right after. For other types of slice, Steps (d) and (e) can be performed in a thread in parallel with Step (f) and beyond.

d. This is for accurate construction of reference pictures. Inverse quantizationnd inverse transformation (**IQT**) are performed and added to the predictions to have the decoded blocks.

e. **ILDB** is applied accordingly to the reconstructed blocks.

f. Meanwhile macroblock codes: including its configuration info (types and modes), motion info (motion vectors and reference ids), and residual info (quantized coefficients), are collected for packing (**BSE**) in the following sub-steps:

    i. Code clean-up (in **MPR**). Check and verify Mbtype and Cbps, use Skip or Zero respectively if one can. In principal, when there are equivalent codes, use the simple one.

    ii. Drop dependency (in **MPR**). Calculate relative codes from the absolute codes by associate thm with neighborhood information. All neighborhood correlations are solved in this step.

    iii. Unify symbols (in **SEC**). Translate relative codes into symbols, and table or context indices that are independent of the concept of syntax type.

    iv. Entropy coding (**VLE**) on symbols.

g. Parsing bitstream data in RBSP form (in **VLE**), and output to application or driver.

h. By the end of each picture, write out the accurate actual data size to designate buffer for ENC to access.

1.2.3 Synchronization Mechanism

Encoding of a video stream can be broken down to three major steps (as explained in the previous section):

1. VPP: video-stream pre-processing.
2. ENC: encoding, *i.e.* code decision of inter-MVs and intra-modes. And
3. PAK: bit-stream packing,

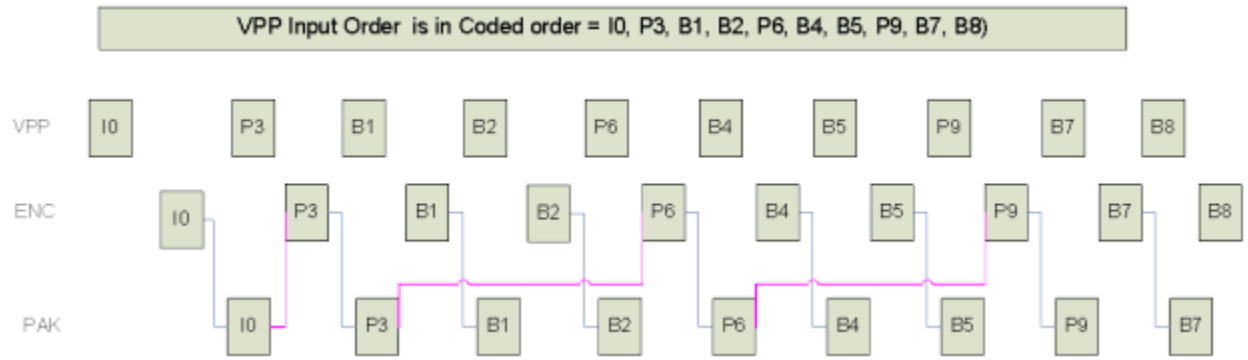
a) residual calculation, transformation, and quantization,
b) code bit-stream packing, and
c) refenrece generation of keyframes.

This section describes an architectural solution to map first two steps in the GFX engine and the last step in the MFX engine. Since this involves two OS visible engines, managing them using in parallel under one application is similar to the solution in BLC/CTG implementations. Each engine has its own command streamers and has mechanisms to synchronize at required level as described in the next sub-section.

Above three steps of encoding have dependencies in processing based on

i. functional pipeline order, *i.e.* on a given frame, VPP needs to be performed first, then ENC, then PAK and finally MFD (*Multi-Format Decoding*) for key reference frame generation.
ii. I-frames are key frames for P and B, they have to be first in every pipe-stage.
iii. P-frames are key frames for B frames and therefore P frames are processed first before the dependent B frames
iv. GFX Engine is time slice to work on either VPP or ENC frame as we discussed in the previous chapter.
v. PAK + MFD are executed on the same frame in the MFX engine by macro-block level pipelining within a slice. It should be noted that for the sake of simplicity, an entire frame (potentially multiple slices) are processed in the corresponding engine and no smaller granularity of switching is allowed between the functional pipeline stages.

Three steps of the encoding can be interleaved on two engines in the following way on a frame by frame basis.



Command Stream Synchronization

1.2.4 Restrictions

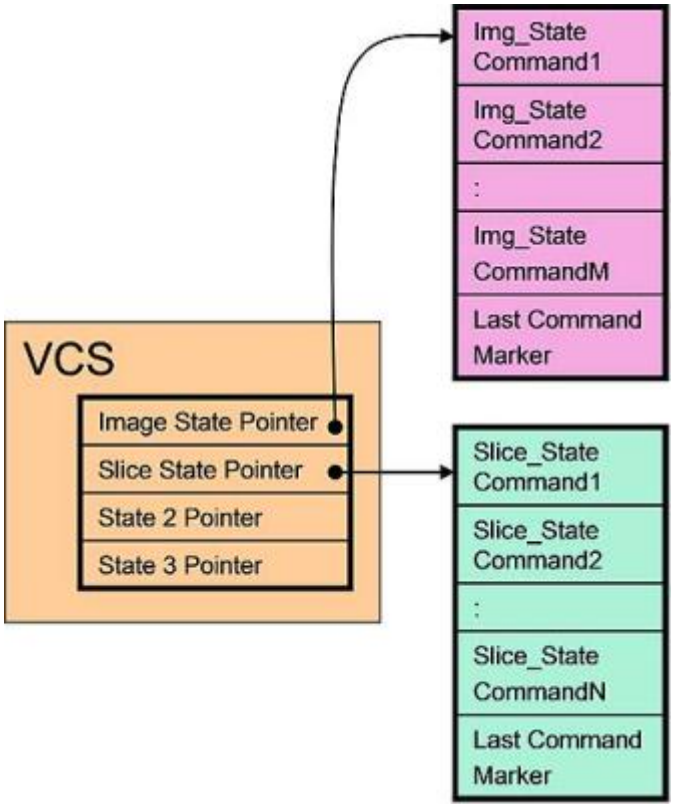
MFC implementation is subject to the following limitations.

- Context switching within MFC and with Graphics Engine occurs only at frame boundary to minimize the amount of information need to be tracked and maintained.

1.3 MFX State Model

The parallel video engine (PVE) supports two state delivery models: inline state model and indirect state model. For inline state model, the state commands (*_STATE) can be issued in batch buffers or ring buffers directly preceding object commands (*_OBJECT). In the indirect state model, the state commands are not placed in the batch buffers or ring buffers. Instead Indirect State Buffers provide state information (in the form of the above mentioned state commands) for the MFX pipeline. The MFX_STATE_POINTER command provides the memory pointer to a indirect state buffer.

VCS (aka BCS) handles the difference of the two state delivery models. Therefore, the MFX pipeline always sees the state commands in both models. However, MFX hardware supports additional context save/restore of 'dynamic states'. Dynamic states are the internal signals that are persistent. This could be the CABAC context for macroblock encoding.



MFX State Model

The MFX codec is designed to be a stateless engine, that it does not retain any history of settings (states) for the encoding/decoding process of a picture. Hence, driver must issue the full set of MFX picture state command sequence prior to process each new picture. In addition, driver must issue the full set of Slice state command sequence prior to process a slice.

- In particular, RC6 always happens between frame boundaries. So at the beginning of every frame, all state information needs to be programmed. There is no state information as part of media context definition.

1.4 MFX Interruptability Model

MFX encoding and the encoding pipeline do not support interruption. All operations are frame based. Interrupts can only occur between frames; the driver will submit all the states at the beginning of each frame. Any state kept across frames is in MMIO registers that should be read between frames.

Software submits without any knowledge of where the parser head pointer is located. Also there is a non-deterministic amount of time for the new context to reach the command streamer. However, the state model for the MFX engine requires software to know exactly what state the pipeline is in at all times. This introduces cases where a preemption could occur during or after a state change without software ever knowing the state saved out to memory on the context switch.

Also, preemption is only allowed during the last macroblock in a row. Hardware cannot always perform a context switch when the new context is seen by the hardware. To avoid a switch during an invalid macroblock and to keep the state synchronized with software, there are two commands available that are used. MI_ARB_ON_OFF disables and enables preemption while MFX_WAIT ensures the context switch, if needed, preempts during macroblock execution. Below illustrates an example assuming VC1 VLD mode.

Command Ring/Batch	Notes
MI_ARB_ON_OFF = OFF	Disable preemption
S1	Inline or indirect state cmd 1
S2	Inline or indirect state cmd 2
S3	Inline or indirect state cmd 3
XXXX_OBJECT	Slice
MI_ARB_ON_OFF = ON	Enable preemption
MFX_WAIT	Allow preemption to occur while XXXX_OBJECT executes
MI_ARB_ON_OFF = OFF	Since arbitration is off again, state commands are allowed below
S4	Inline or indirect state cmd 4
S5	Inline or indirect state cmd 5
S6	Inline or indirect state cmd 6
XXXX_OBJECT	Slice
MI_ARB_ON_OFF = ON	Enable preemption
MFX_WAIT	Allow preemption to occur while XXXX_OBJECT executes
MI_ARB_ON_OFF = OFF	Since arbitration is off again, state commands are allowed below

Note that store DW commands may execute inside the preemption enabling window if needed.

1.5 MFX Programming Restrictions

1.5.1 All Codecs

There is a hardware issue to switch to JPEG decode if the last MB of the previous video frame (AVC/VC1/MPEG decode or AVC encode) has no coefficients coded (CBP equals 0).

To resolve the above issue, an AVC frame with only 1x1 intra-coded MB must be issued before JPEG frame. Both AVC frame and JPEG frame must be placed in the same batch buffer to ensure JPEG frame is executed immediately after the added AVC frame.

The alternate WA is to place the AVC 1x1 intra-coded MB frame immediately after any AVC/VC1/MPEG decode or AVC encode frame.

1.6 MFX Codec Commands Summary

DWord	Bit	Description
0	31:29	**Instruction Type =** GFXPIPE = 3h
	28:16	**3D Instruction Opcode =** PIPELINE_SELECT GFXPIPE[28:27 = 1h, 26:24 = 1h, 23:16 = 04h] (Single DW, Non-pipelined)
	15:1	**Reserved**: MBZ
	0	**Pipeline Select** 0: 3D pipeline is selected 1: Media pipeline is selected

Pipeline Type (28:27)	Opcode (26:24)	Sub Opcode (23:16)	Command	Definition Chapter
VC1 State				
2h	5h	0h	VC1_BSD_PIC_STATE	VC1 BSD
2h	5h	1h	Reserved	n/a
2h	5h	2h	Reserved	n/a
2h	5h	3h	VC1_BSD_BUF_BASE_STATE	VC1 BSD
2h	5h	4h	Reserved	n/a
2h	5h	5h-7h	Reserved	n/a
VC1 Object				
2h	5h	8h	VC1_BSD_OBJECT	VC1 BSD
2h	5h	9h-FFh	Reserved	n/a

Pipeline Type (28:27)	Opcode (26:24)	Sub Opcode (23:16)	Command	Definition Chapter
State				
2h	6h	0h		GPU Overview
2h	6h	9h		GPU Overview
2h	6h	2h-7h	Reserved	n/a
Object				
2h	6h	8h		GPU Overview
2h	6h	9h-FFh	Reserved	n/a

Note that it is possible for a command to appear in both IMAGE and SLICE state buffer, e.g. QM_STATE for JPEG can be issued at frame level or scan/slice level.

Pipeline Type (28:27)	Opcode (26:24)	SubopA (23:21)	Subop B (20:16)	Command	Chapter	Recommended Indirect State Pointer Map	Interruptable?
	MFX Common	Common					
2h	0h	0h	0h	MFX_PIPE_MODE_SELECT	MFX	IMAGE	No
2h	0h	0h	1h	MFX_SURFACE_STATE	MFX	IMAGE	No
2h	0h	0h	2h	MFX_PIPE_BUF_ADDR_STATE	MFX	IMAGE	No
2h	0h	0h	3h	MFX_IND_OBJ_BASE_ADDR_STATE	MFX	IMAGE	No
2h	0h	0h	4h	MFX_BSP_BUF_BASE_ADDR_STATE	MFX	IMAGE	No
2h	0h	0h	6h	MFX_ STATE_POINTER	MFX	IMAGE	No
2h	0h	0h	7h	MFX_QM_STATE	MFX	IMAGE/SLICE	No
2h	0h	0h	8h	MFX_FQM_STATE	MFX	IMAGE	No
2h	0h	0h	A-1Eh	Reserved	n/a	n/a	No
2h	0h	0h	1FH	MFX_muC_IND_OBJ_BASE_ADDR_STATE	MFX	IMAGE	No
	MFX Common	Dec					
2h	0h	1h	0-8h	Reserved	n/a	n/a	n/a
2h	0h	1h	9h	MFD_ IT_OBJECT	MFX	n/a	No
2h	0h	1h	A-1Fh	Reserved	n/a	n/a	n/a
	MFX Common	Enc					
2h	0h	2h	0-7Fh	Reserved	n/a	n/a	n/a
2h	0h	2h	8h	MFX_PAK_INSERT_OBJECT	MFX	n/a	No
2h	0h	2h	9h	Reserved	n/a	n/a	n/a
2h	0h	2h	Ah	MFX_STITCH_OBJECT	MFX	n/a	No
2h	0h	2h	B-1Fh	Reserved	n/a	n/a	n/a
	AVC/ MVC	Common (State)					
2h	1h	0h	0h	MFX_AVC_IMG_STATE	MFX	IMAGE	n/a
2h	1h	0h	1h	Reserved	n/a	n/a	n/a
2h	1h	0h	2h	MFX_AVC_DIRECTMODE_STATE	MFX	SLICE	n/a
2h	1h	0h	3h	MFX_AVC_SLICE_STATE	MFX	SLICE	n/a
2h	1h	0h	4h	MFX_AVC_REF_IDX_STATE	MFX	SLICE	n/a
2h	1h	0h	5h	MFX_AVC_WEIGHTOFFSET_STATE	MFX	SLICE	n/a
	AVC/ MVC	Dec					
2h	1h	1h	0-5h	Reserved	MFX	n/a	n/a
2h	1h	1h	6h	MFD_AVC_DPB_STATE	MFX	IMAGE	n/a
2h	1h	1h	7h	MFD_AVC_SLICEADDR_OBJECT	MFX	n/a	n/a
2h	1h	1h	8h	MFD_AVC_BSD_OBJECT	MFX	n/a	No
2h	1h	1h	9-1Fh	Reserved	n/a	n/a	n/a
	AVC/ MVC	Enc					
2h	1h	2h	0-8h	Reserved	n/a	n/a	n/a
2h	1h	2h	9h	MFC_AVC_PAK_OBJECT	MFX	n/a	No
2h	1h	2h	A-1Fh	Reserved	n/a	n/a	n/a
	AVC/ MVC	Extension					

	VC1	Common (State)					
2h	2h	0h	0h	Reserved	n/a	n/a	n/a
2h	2h	0h	1h	MFX_VC1_PRED_PIPE_STATE	MFX	IMAGE	n/a
2h	2h	0h	2h	MFX_VC1_DIRECTMODE_STATE	MFX	SLICE	n/a
2h	2h	0h	3-1Fh	Reserved	n/a	n/a	n/a
	VC1	Dec					
2h	2h	1h	0h	MFD_VC1_SHORT_PIC_STATE	MFX	IMAGE	n/a
2h	2h	1h	1h	MFD_VC1_LONG_PIC_STATE	MFX	IMAGE	n/a
2h	2h	1h	2-7h	Reserved	n/a	n/a	n/a
2h	2h	1h	8h	MFD_VC1_BSD_OBJECT	MFX	n/a	No
2h	2h	1h	9-1Fh	Reserved	n/a	n/a	n/a
	VC1	Enc					
2h	2h	2h	0-1Fh	Reserved	n/a	n/a	n/a
	MPEG2	Common (State)					
2h	3h	0h	0h	MFX_MPEG2_PIC_STATE	MFX	IMAGE	n/a
2h	3h	0h	1-1Fh	Reserved	n/a	n/a	n/a
	MPEG2	Dec					
2h	3h	1h	1-7h	Reserved	n/a	n/a	n/a
2h	3h	1h	8h	MFD_MPEG2_BSD_OBJECT	MFX	n/a	No
2h	3h	1h	9-1Fh	Reserved	n/a	n/a	n/a
	MPEG2	Enc					
2h	3h	2h	0-2h	Reserved	n/a	n/a	n/a
2h	3h	2h	3-8h	Reserved			
2h	3h	2h	9h	MFC_MPEG2_SLICEGROUP_STATE			
2h	3h	2h	A-1Fh	Reserved			
	MuC	Common (State)					
2h	5h	0h		Reserved			
	MuC	Enc					
2h	5h	2h		Reserved			
	JPEG	Common					
2h	7h	0h	0h	MFX_JPEG_PIC_STATE	MFX	IMAGE	No
2h	7h	0h	1h	Reserved	n/a	n/a	n/a
2h	7h	0h	2h	MFX_JPEG_HUFF_TABLE_STATE	MFX	IMAGE	No
2h	7h	0h	3-1Fh	Reserved	n/a	n/a	n/a
	JPEG	Dec					
2h	7h	1h	1-7h	Reserved	MFX	n/a	n/a
2h	7h	1h	8h	MFD_JPEG_BSD_OBJECT	MFX	MCU	No
2h	7h	1h	9-1Fh	Reserved	MFX	n/a	n/a
	JPEG	Enc					
2h	7h	2h	0-1Fh	Reserved	MFX	n/a	n/a

MMIO Space Registers

Range Start	Range End	Unit owner
00002000	00002FFF	Render/Generic Media Engine
00004000	00004FFF	Render/Generic Media Graphics Memory Arbiter
00006000	00007FFF	
00012000	000123FF	MFX Control Engine (Video Command Streamer)
00012400	00012FFF	Media Units (VIN unit)
00014000	00014FFF	MFX Memory Arbiter
00022000	00022FFF	Blitter Engine
00024000	00024FFF	Blitter Memory Arbiter
00030000	0003FFFF	
00100000	00107FFF	Fence Registers
00140000	0017FFFF	MCHBAR (SA)

Memory Interface Command Map

04h Opcode (28:23)	MI_FLUSH

1.6.1 MFX Decoder Commands Sequence

The MFX codec is designed to be a stateless engine, that it does not retain any history of settings (states) for the encoding/decoding process of a picture. Hence, driver must issue the full set of MFX picture state command sequence prior to process each new picture. In addition, driver must issue the full set of Slice state command sequence prior to process a slice.

In particular, RC6 always happens between frame boundaries. So at the beginning of every frame, all state information needs to be programmed. There is no state information as part of media context definition

1.6.1.1 Examples for AVC

The following gives a sample command sequence programmed by a driver

a) For Intel or DXVA2 AVC Long Slice Bitstream Format

MFX_PIPE_MODE_SELECT

MFX_SURFACE_STATE

MFX_PIPE_BUF_ADDR_STATE

MFX_IND_OBJ_BASE_ADDR_STATE

MFX_BSP_BUF_BASE_ADDR_STATE

MFX_QM_STATE

VLD mode: MFX_AVC_PICID_STATE

MFX_AVC_IMG_STATE

MFX_AVC_DIRECTMODE_STATE

MFX_AVC_REF_IDX_STATE

MFX_AVC_WEIGHTOFFSET_STATE

MFX_AVC_SLICE_STATE

VLD mode: MFD_AVC_BSD_OBJECT

IT mode: MFD_IT_OBJECT

MI_FLUSH

b) For DXVA2 AVC Short Slice Bitstream Format (for VLD mode only)

MFX_PIPE_MODE_SELECT

MFX_SURFACE_STATE

MFX_PIPE_BUF_ADDR_STATE

MFX_IND_OBJ_BASE_ADDR_STATE

MFX_BSP_BUF_BASE_ADDR_STATE

MFD_AVC_DPB_STATE

VLD mode: MFX_AVC_PICID_STATE

MFX_AVC_IMG_STATE

MFX_QM_STATE

MFX_AVC_DIRECTMODE_STATE

MFX_AVC_REF_IDX_STATE

MFX_AVC_WEIGHTOFFSET_STATE

VLD mode : MFD_AVC_SLICEADDR_OBJECT

VLD mode: MFD_AVC_BSD_OBJECT

VLD mode : MFD_AVC_BSD_SLICEADDR_OBJECT

VLD mode: MFD_AVC_BSD_OBJECT

… repeat these four commands N-1 times for a N-slice picture

VLD mode: MFD_AVC_BSD_OBJECT (for the last slice of the picture)

MI_FLUSH

1.6.1.2 Examples for VC1

The following gives a sample command sequence programmed by a driver

a) For Intel Proprietary Long Bitstream Format

MFX_VC1_DIRECTMODE_STATE

MFX_VC1_PRED_PIPE_STATE

MFX_VC1_LONG_PIC_STATE

VLD mode: MFD_VC1_BSD_OBJECT

IT mode: MFD_IT_OBJECT

MI_FLUSH

b) For DXVA2 VC1 Compliant Bitstream Format (for VLD mode only)

MFX_VC1_DIRECTMODE_STATE

MFX_VC1_PRED_PIPE_STATE

MFX_VC1_SHORT_PIC_STATE

VLD mode: MFD_VC1_BSD_OBJECT

MI_FLUSH

c) For DXVA2 VC1 Compliant Bitstream Format (for VLD mode only), and field pair picture

Batch buffer for top-field

states....

Slice_objs...

MI_flush

store register immediate (if VC1 short format with interlaced field pic)

MI_flush

Batch buffer for bottom field

load register immediate (if VC1 short format with interlaced field pic)

MI_flush

states....

Slice_objs...

MI_flush

1.6.1.3 Examples for JPEG

The following gives a sample command sequence programmed by a driver

Programmed once at the start of decoding

MFX_PIPE_MODE_SELECT

MFX_PIPE_SURFACE_STATE

MFX_IND_OBJ_BASE_ADDR_STATE

MFX_PIPE_BUF_ADDR_STATE

MFX_JPEG_PIC_STATE

Programmed at the start of Frame or Scan (These commands can be sent multiple times either before MFX_JPEG_PIC_STATE or before MFD_JPEG_BSD_OBJECT)

MFX_JPEG_HUFF_TABLE

MFX_QM_STATE

Programmed per Scan (These commands can be sent multiple times depending on each bit stream)

MFD_JPEG_ BSD_OBJECT

MI_FLUSH

1.7 MFX Pipe Common Commands

1.7.1 MFX_WAIT Command

This is a VCS Command to synchronize the two concurrent pipeline (one is VCR and one if MFX). It is included here, because it is only used by MFX pipeline. It defines the usage model for pre-emption which makes it very easy to do any ring buffer programming.

VCS Command engine services both VCR and MFX pipelines. If VCR takes many clock to initialize , and if no WAIT command, MFX Object Command can start sending data to VCR before it is initialized. Hence, this command effective causes a flush in the selected pipeline (VCR or MFX).

MFX_WAIT

Source:	VideoCS
Length Bias:	1

This command can be considered the same as an MI_NOOP except that the command parser will not parse the next command until the following happens

- **AVC or VC1 BSD mode:** The command will stall the parser until completion of the BSD object
- **IT, encoder, and MPEG2 BSD mode:** The command will stall the parser until the object package is sent down the pipelineThis command should be used to ensure the preemption enable window occurs during the time the object command is being executed down the pipeline.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 03h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Command Subtype** Default Value: 01h MFX_SINGLE_DW Format: OpCode
	26:16	**Sub-Opcode** Default Value: 0h MFX_WAIT Format: OpCode
	15:10	**Reserved** Project: All Format: MBZ
	8	**MFX Sync Control Flag** If set, VCS will stall the parser until all prior MFX objects are completed down the MFX pipeline
	7:6	**Reserved** Project: All Format: MBZ
	5:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Project: All Format: =n Total Length - 2

1.7.2 MFX_STATE_POINTER Command

MFX_STATE_POINTER	
Project:	All
Source:	VideoCS
Length Bias:	2

The MFX_STATE_POINTER command, issued at picture level, is used to set up the indirect pointers for VCS to fetch all the MFX states (Image state, Slice state, etc.) needed for the encoding/decoding process in PAK/IT mode. The encoding/decoding states are presented by state commands, which are grouped into separate sets (picture level, slice level, etc.), and each is stored in its own memory buffer referred by an indirect state pointer. The content of each indirect state buffer is a list of MFX state commands with no special format requirements. The sequence of commands in each indirect state buffer is terminated by a MI_BATCH_BUFFER_END command (acts as the last command marker). Therefore, indirect state buffers can have different and variable length of command sequences.

The indirection is designed to facilitate context switching in the middle of a codec operation. The smallest granularity of interruption is designed to be at a completed MB row in AVC/VC1/MPEG2 IT and AVC PAK operating modes as well as in VC1/MPEG2 VLD mode. There is no support for context switch in AVC VLD mode.

Hardware supports up to 4 separate indirect state pointers, allowing software to manage the grouping of state commands. During context switch, hardware restores (re-issues) the latest version of each indirect state pointer, if present.

MFX_STATE_POINTER command can only program one indirect state pointer at a time. MI_FLUSH will invalidate all indirect state buffer pointers inside VCS.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFX_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:24	**Media Command Opcode** Default Value: 0h MFX_COMMON_STATE Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 6h Format: OpCode
	15:12	**Reserved** Project: All Format: MBZ
	11:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Project: All Format: =n Total Length - 2
1	31:5	**State Pointer**

MFX_STATE_POINTER

		Format: GeneralStateOffset[31:5]Indirect State Buffer
		Specifies the 32-byte aligned address of an Indirect State Buffer. This pointer is relative to the General State Base Address.
	4:2	**Reserved**
		Project: All
		Format: MBZ
	1:0	**State Pointer Index**
		Specifies one of the four indirect state pointers to program.

Value	Name	Description	Project
00b		indirect state pointer 0 (image state)	All
01b		indirect state pointer 1 (slice state)sc	All
10b		indirect state pointer 2	
11b		indirect state pointer 3	

1.7.3 MFX_PIPE_MODE_SELECT

MFX_PIPE_MODE_SELECT

Source:	VideoCS
Length Bias:	2

Specifies which codec and hardware module is being used to encode/decode the video data, on a per-frame basis.

The MFX_PIPE_MODE_SELECT command specifies which codec and hardware module is being used to encode/decode the video data, on a per-frame basis. It also configures the hardware pipeline according to the active encoder/decoder operating mode for encoding/decoding the current picture. Commands issued specifically for AVC and MPEG2 are ignored when VC1 is the active codec.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_COMMON Format: OpCode
	26:24	**Opcode** Default Value: 0h MFX_COMMON_STATE Format: OpCode
	23:21	**SubOpA** Default Value: 0h Format: OpCode
	20:16	**SubOpB** Default Value: 0h MFX_PIPE_MODE_SELECT Format: OpCode
	15:12	**Reserved** Project: All Format: MBZ
	11:0	**DWord Length** Project: All Format: =n Total Length - 2 Value / Name / Description 2h / DWORD_COUNT_n **[Default]** / Excludes DWord (0,1)
1	26:25	**Reserved** Project: All Format: MBZ
	23:18	**Reserved** Project: All Format: MBZ
	17	**Decoder Short Format Mode** For IT mode, this bit must be 0.

MFX_PIPE_MODE_SELECT

Value	Name	Description
1	Long Format Driver Interface	AVC/VC1/MVC Long Format Mode is in use
0	Short Format Driver Interface **[Default]**	AVC/VC1/MVC Short Format Mode is in use

16:15 **Decoder Mode select**

Each coding standard supports two entry points: VLD entry point and IT (IDCT) entry point. This field selects which one is in use.This field is only valid if Codec Select is 0 (decoder).

Value	Name	Description
0h	VLD Mode	All codec minimum must support this modeConfigure the MFD Engine for VLD Mode Note: All codec minimum must support this mode
1h	IT Mode	Configure the MFD Engine for IT Mode Note: Only VC1 and MPEG2 support this mode

14:13 **Reserved**

Project:	All
Format:	MBZ

12 **Reserved**

Format:	MBZ

11 **Pic Error/Status Report Enable.**

This field control whether the error/status reporting is enable or not.0: Disable1: EnableIn decoder modes: Error reporting is written out once per frame. The Error Report frame ID listed in DW3 along with the VLD/IT error status bits are packed into one cache and written to the “Decoded Picture Error/Status Buffer address” listed in the MFX_PIPE_BUF_ADDR_STATE Command. Note: driver shall program different error buffer addresses between pictrues; otherwise, hardware might overwrite previous written data if driver does not read it fast enough.In encoder modes: Not used

Value	Name
0h	Disable
1h	Enable

10 **Stream-Out Enable**

This field controls whether the macroblock parameter stream-out is enabled during VLD decoding for transcoding purpose.

Value	Name
0h	Disable
1h	Enable

Programming Notes

In decoder modes: The Stream-Out feature is added to support transcoding. While decoding the input compressed stream, selected decoded information may be used by the encoder for re-compression.In encoder modes: This feature used to perform dynamic Multipass of PAK for conformance pupose. Also it provides feedback to host (ENC) for future needs. Software can use this bit to disable writing PAK steam data to the streamout buffer for last pass of frame in PAK. Thus, save memory bandwidth.

9 **Post Deblocking Output Enable (PostDeblockOutEnable)**

Project:	All

This field controls the output write for the reconstructed pixels AFTER the deblocking filter.In MPEG2 decoding mode, if this is enabled, VC1 deblocking filter is used.

Value	Name

0h	Disable	
1h	Enable	

8 — Pre Deblocking Output Enable (PreDeblockOutEnable)

Project: All

This field controls the output write for the reconstructed pixels BEFORE the deblocking filter.

Value	Name
0h	Disable
1h	Enable

7:6 — Reserved

Format: MBZ

5 — Stitch Mode

Project: All

Exists If: CodecSel=Encode and StandardSel=AVC

Value	Name	Description
0h	Not in stitch mode	
1h	In the special stitch mode	This mode can be used for any Codec as long as bitfield conditions are met.

4 — Codec Select

Value	Name	Description
0h	Decode	
1h	Encode	Valid only if StandardSel is AVC, MPEG2)

3:0 — Standard Select

Value	Name	Description
0000b	MPEG2	
0001b	VC1	
0010b	AVC	Covers both AVC and MVC
0011b	JPEG	
0110b	Reserved	
0111b	Reserved	

2 — 31:11 Reserved

Format: MBZ

10 — MPC pref08x8_disable Flag (Default 0)

BitFieldDesc

Value	Name	Project
0h	Disable	All
1h	Enable	All

9:8 — Reserved

Format: MBZ

6 — Clock gate Enable at Slice-level

BitFieldDesc:

Value	Name	Description
0h	Disable	Disable Slice-level Clock gating, Unit-level Clock gating will apply
1h	Enable	Enable Slice-level Clock gating, overrides any Unit level Clock gating

4 — AVC Motion Vector/POC Table Error Disable Flag

This bit disable termination due to any errors resulting from Motion Vector and POC Table

Value	Name	Description

MFX_PIPE_MODE_SELECT

		0h	Terminates	Motion Vector/POC Table Error will terminate the current slice decoding
		1h	Will not terminate	Motion Vector/POC Table Error will not terminate the current slice decoding
	3	**AVC Mbdata Error Disable Flag** This bit disable termination due to any errors resulting from any Mbdata (QP delta range)		
		Value	**Name**	**Description**
		0h	Enable	Mbdata Error will terminate the current slice decoding
		1h	Disable	Mbdata Error will not terminate the current slice decoding
	2	**AVC CABAC/CAVLC Decode Error Disable Flag** This bit disable termination due to any errors resulting from CABAC/CAVLC decoding engine		
		Value	**Name**	**Description**
		0h	Terminate	CABAC/CAVLC Decoder Error will terminate the current slice decoding
		1h	Will not terminate	CABAC/CAVLC Decoder Error will not terminate the current slice decoding
	1	**Reserved** Format: MBZ		
	0	**Reserved** Format: MBZ		
3	31:0	**Pic Status/Error Report ID** Format: U32 In decoder modes: Error reporting is written out once per frame. This field along with the VLD error status bits are packed into one cache and written to the memory location specified by “Decoded Picture Error/Status Buffer address” listed in the MFX_PIPE_BUF_ADDR_STATE Command. In encoder modes: Not used		
		Value	**Name**	**Description**
		0h	32-bit unsigned	Unique ID Number
		1h	Reserved	
4	31:0	**Reserved** Format: MBZ		

The Encoder Pipeline Modes of Operation (Per Frame):

1. PAK Mode: VCS-command driven, setup by driver. Like the IT mode of decoder, it is executed on a per-MB basis. Hence, each PAK Object command corresponds to coding of only one MB.
    a. Normal Mode (including transcoding): receive per-MB control and data (MV, mb_type, cbp, etc.). It generates the output compressed bitstream as well as the reconstructed reference pictures, one MB at a time, for later use.
    b. Encoder StreamOut Mode: to provide per-MB, per-Slice and per-Frame coding result and information (statistics) to the Host, Video Preprocessing Unit and ENC Unit to enhance their operations.

The Decoder Pipeline Modes of Operation (Per Frame):

1. VLD Mode: The output from the BSD (weight&offset/coeff/motion vectors record) can be sent in part (as specified) and to the remaining fixed function hardware pipeline to complete the decoding processing. The driver specifies through MFD commands of what to send out from the BSD unit and where to send the BSD output.
    a. For transcoding (including transrating and transcaling), part of the BSD output (a series of per-MB record) can be sent to memory for further processing to encode into

a difference output format. This function is named as StreamOut. When StreamOut is active, not all MB information needs to be sent, only MVs and selective MB coding information.

2. IT Mode: In this mode, the BSD is not invoked. Instead host performs all the bitstream decoding and parsing; and the result are saved into memory in a specific per-MB record format. The MFD Engine VCS reads in these records one at time and finish the rest of the decoding (IT, MC, IntraPred and ILDB).
    a. MB information is organized into two indirect data buffers, one for MVs and one for residue coefficients. As such, two indirect base address pointers are defined.

Programming Restriction:

- Software must ensure the current pipeline is flushed via an MI_FLUSH prior to the execution of MFX_PIPE_MODE_SELECT in switching the MFX Engine to encode/decode a different codec format (AVC, VC1 or MPEG2).
- MFX_PIPE_MODE_SELECT is issued per picture (frame or field).

Emulation Prevention Byte Removal is handled in 2 different ways that affects the definition of the Slice Data Buffer and H/W behavior. A control mode bit is defined to switch between these 2 methods.

1. Application is required to remove any emulation prevention byte straddling across the Slice Header and Slice Data boundary in the bitstream
    a. As such, application will pass to the driver the exact starting location of Slice Data (Byte Offset and Bit Offset) in the Slice buffer and gurantee there is no Emulation Prevention Byte at the beginning of Slice Data. So H/W does not need to do any before the Slice Data.
2. Application does not remove any emulation prevention byte straddling across the Slice Header and Slice Data boundary in the bitstream
    a. As such application will pass to the driver the same set of information as above, but H/W is now required to scan the Emulation Prevention Byte from the beginning of Slice Header through to Slice Data, in order to locate the exact starting point of the Slice Data.
        i. This is because the Slice Header Byte Offset, as defined in DXVA2 interface, does not include any emulation bytes found in the Slice Header.

1.7.4 MFX_SURFACE_STATE Command

MFX_SURFACE_STATE	
Source:	VideoCS
Length Bias:	2

This command is common for all encoding/decoding modes, to specify the uncompressed YUV picture (i.e. destination surface) or intermediate streamout in/out surface (e.g. coefficient/residual) (field, frame or interleaved frame) format for reading and writing:

- Uncompressed, original input picture to be encoded
- Reconstructed non-filtered/filtered display picturec(becoming reference pictures as well for subsequent temporal inter-prediction)

Since there is only one media surface state being active during the entire encoding/decoding process, all the uncompressed/reconstructed pictures are defined to have the same surface state. The primary difference among picture surface states is their individual programmed base addresses, which are provided by other state commands and not included in this command. MFX engine is making the association of surface states and corresponding buffer base addresses.

MFX engine currently supports only one media surface type for video and that is the NV12 (Planar YUV420 with interleaved U (Cb) and V (Cr). For optimizing memory efficiency based on access patterns, only TileY is supported. For JPEG decoder, only IMC1 and IMC3 are supported. Pitch can be wider than the Picture Width in pixels and garbage will be there at the end of each line. The following describes all the different formats that are supported and not supported in Gen7 MFX :

- NV12 – 4:2:0 only; UV interleaved; Full Pitch, U and V offset is set to 0 (the only format supported for video codec); vertical UV offset is MB aligned; UV xoffsets = 0. JPEG does not support NV12 format because non-interleave JPEG has performance issue with partial write (in interleaved UV format)
- IMC 1 & 3 – Full Pitch, U and V are separate plane; (JPEG only; U plane + garbage first in full pitch followed by V plane + garbage in full pitch). U and V vertical offsets are block aligned; U and V xoffset = 0; there is no gap between Y, U and V planes. IMC1 and IMC3 are different by a swap of U and V. This is the only format supported in JPEG for all video subsampling types (4:4:4, 4:2:2 and 4:2:0)
- We are not supporting IMC 2 & 4 – Full Pitch, U and V are separate plane (JPEG only; U plane first in full pitch followed by V plane in full pitch – U and V plane are side-by-side). U and V vertical offsets are 16-pixel aligned; V xoffset is half-pitch aligned; U xoffset is 0; there is no gap between Y, U and V planes. IMC2 and IMC4 are different by a swap of U and V.
- We are not supporting YV12 – half pitch for each U and V plane, and separate planes for Y, U and V (U plane first in half pitch followed by V plane in half pitch). For YV12, U and V vertical offsets are block aligned; U and V xoffset = 0; there is no gap between Y, U and V planes

Note that the following data structures are not specified through the media surface state

- 1D buffers for row-store and other miscellaneous information.
- 2D buffers for per-MB data-structures (e.g. DMV biffer, MB info record, ILDB Control and Tcoeff/Stocoeff).

This surface state here is identical to the Surface State for deinterlace and sample_8x8 messages described in the Shared Function Volume and Sampler Chapter.

For non pixel data, such as row stores, indirect data (Compressed Slice Data, AVC MV record, Coeff record and AVC ILDB record) and streamin/out and output compressed bitstream, a linear buffer is employed. For row stores, the H/W is designed to guarantee legal memory accesses (read and write). For the remaining cases, indirect object base address, indirect object address upper bound, object data start address (offset) and object data length are used to fully specified their corresponding buffer. This mechanism is chosen over the pixel surface type because of their variable record sizes.

MFX_SURFACE_STATE

All row store surfaces are linear surface. Their addresses are programmed in Pipe_Buf_Base_State or Bsp_Buf_Base_Addr_State

Programming Notes

VC1 I picture scaling: Even though VC1 allows I reconstructed picture scaling (via RESPIC), as such scaling is only allowed at I picture. All subsequent P (and B) pictures must have the same picture dimensions with the preceding I picture. Therefore, all reference pictures for P or B picture can share the same surface state with the current P and B picture. Note : H/W is not processing RESPIC. Application is no longer expecting intel decoder pipelineand kernel to perform this function, it is going to be done in the video post-processing scaler or display controller scale as a separate step and controller.

All video codec surfaces must be NV12 Compliant, except JPEG. U/V vertical must be MB aligned for all video codec (further contrained for field picture), but JPEG can be block aligned. All video codec and JPEG uses Tiled – Y format only, for uncompressed pixel surfaces.

Even for JPEG planar 420 surface, application may provide only 1 buffers, but there is still only one single surface state for all of them. If IMC equal to 1, 2, 3 or 4, U and V have the pitch same as Y. And U and V will have different offset, each offset is block aligned.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_COMMON Format: OpCode
	26:24	**Opcode** Default Value: 0h MFX_COMMON_STATE Format: OpCode
	23:21	**SubOpA** Default Value: 0h Format: OpCode
	20:16	**SubOpB** Default Value: 1h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Total Length - 2 Value / Name / Description 4h / DWORD_COUNT_n **[Default]** / Excludes DWord (0,1)
1	31:2	**Reserved** Format: MBZ
	1:0	**Reserved** Format: MBZ
2	31:18	**Height** Format: U14-1 Height

This field specifies the height of the Picture in units of pixels/residuals. For PLANAR surface formats, this field indicates the height of the Y (luma) plane. Note : Gen7 Video Codecs must program less than and equal to 4K.(In future, it will be ideal to have this field define in a WORD boundary.)AVC – multiple of 2 MB rows for field pictureVC1 – mulitple of 4 pixels for field pictureMPEG2 - multiple of 2 MB rows for field picJPEG – mulitple of integral MCU (8 or 16 pixels) per picture

Value	Name	Description
[0,16383]		representing heights [1,16384]

Programming Notes	Project
For AVC : For frame picture is a multiple of 16; for field picture is a multiple of 32 For VC1 : For progressive frames, the frame height and frame width is a multiple of 2 pixels. For interlaced frames, the frame height shall be a multiple of 4 pixels, and its width is a multiple of 2 pixels, based on a PLANAR_420 surface.	
Video Codecs must program less than and equal to 4K. In future, it will be ideal to have this field define in a WORD boundary.	

17:4 **Width**

Format:	U14-1 Width

This field specifies the width of the Picture in units of pixels/residuals. For PLANAR surface formats, this field indicates the width of the Y (luma) plane.

Value	Name	Description
[0,16383]		representing widths [1,16384]

Programming Notes	Project
The Width specified by this field multiplied by the pixel size in bytes must be less than or equal to the surface pitch (specified in bytes via the Surface Pitch field). Width (field value + 1) must be a multiple of 2 for PLANAR_420, MFX HW does not use this field, the picture width is read from IMG State instead, because this field may not equal to the actual picture width. This field is used by the KMD to allocate surface in GTT.	
Video Codecs must program less than and equal to 4K. In future, it will be ideal to have this field define in a WORD boundary.	

3:2 **Reserved**

Format:	MBZ

1:0 **Cr(V)/Cb(U) Pixel Offset V Direction**

Project:	All
Format:	U0.2 exactly as shown in the original spec

MFX_SURFACE_STATE

		Specifies the distance to the U/V values with respect to the even numbered Y channels in the V direction
		Programming Notes
		This field is ignored for all formats except PLANAR_420_8
3	31:28	**Surface Format** Specifies the format of the surface. All of the Y and G channels will use table 0 and all of the Cr/Cb/R/B channels will use table 1.Usage: For 420 planar YUV surface, use 4; for monochrome surfaces, use 12. For monochrome surfaces, hardware ignores control fields for Chroma planes.This field must be set to 4 - PLANAR_420_8, or 12 – Y8_UNORMNot used for MFX, and is ignored. But for JPEG decoding, this field should be programmed to the same format as JPEG_PIC_STATE. For video codec, it should set to 4 always.

Value	Name	Description
0	YCRCB_NORMAL	
1	YCRCB_SWAPUVY	
2	YCRCB_SWAPUV	
3	YCRCB_SWAPY	
4	PLANAR_420_8	(NV12, IMC1,2,3,4, YV12)
5	PLANAR_411_8	Deinterlace Only
6	PLANAR_422_8	Deinterlace Only
7	STMM_DN_STATISTICS	Deinterlace Only
8	R10G10B10A2_UNORM	Sample_8x8 Only
9	R8G8B8A8_UNORM	Sample_8x8 Only
10	R8B8_UNORM (CrCb	Sample_8x8 Only
11	R8_UNORM (Cr/Cb)	Sample_8x8 Only
12	Y8_UNORM	Sample_8x8 Only
13,15	Reserved	

	27	**Interleave Chroma** Format: Enable This field indicates that the chroma fields are interleaved in a single plane rather than stored as two separate planes. This field is only used for PLANAR surface formats.For AVC/VC1/MPEG VLD and IT modes : set to Enable to support interleave U/V only.For JPEG : set to Disable for all formats (including 4:2:0) – because JPEG does not support NV12. (This field is needed only if JPEG will support NV12; otherwise is ignored.)

Value	Name
1	Enable
0	Disable

	26	**Reserved** Format: MBZ
	25:22	**Surface Object Control State (MEMORY_OBJECT_CONTROL_STATE)** This 4-bit field is used in various state commands and indirect state objects to define LLC cacheability including graphics data type attributes for memory objects.

Value	Name	Description
2	Graphics Data Type (GFDT)	This field contains the GFDT bit for this surface when writes occur. GFDT can also be set by the GTT. The effective GFDT is the logical OR of this field with the GFDT from the GTT entry. This field is ignored for reads.Format = U1
1	Cacheability Control	This field controls cacheability in the last-level cache (LLC).Format = U2 enumerated type00: use cacheability control bits from GTT entry01: data is not

		cached in LLC 1x: data is cached in LLC

Programming Notes

This field is ignored; H/W uses those values programmed in each of the Buf Address State entries instead.

21:20 **Reserved**

Format:	MBZ

19:3 **Surface Pitch**

Format:	U17-1 pitch in Bytes

This field specifies the surface pitch in (#Bytes).

Value	Name	Description
[0,2047]		to [1B, 2048B]

Programming Notes

For tiled surfaces, the pitch must be a multiple of the tile width (i.e.128 bytes aligned). If Half Pitch for Chroma is set, this field must be a multiple of two tile widths for tiled surfaces, or a multiple of 2 bytes for linear surfaces.For Y-tiled surfaces: Range = [127, 524287] to [128B,256KB] = [1 tile, 2048 tiles]

2 **Half Pitch for Chroma**

Format:	Enable

(This field must be set to Disable)This field indicates that the chroma plane(s) will use a pitch equal to half the value specified in the Surface Pitch field. This field is only used for PLANAR surface formats.This field is igored by MFX (unless we support YV12)

1 **Tiled Surface**

Format:	Boolean

(This field must be set to TRUE: Tiled)This field specifies whether the surface is tiled.This field is ignored by MFX

Value	Name	Description
0	False	Linear
1	True	Tiled

Programming Notes

Linear surfaces can be mapped to Main Memory (uncached) or System Memory (cacheable, snooped). Tiled surfaces can only be mapped to Main Memory.The corresponding cache(s) must be invalidated before a previously accessed surface is accessed again with an altered state of this bit.

0 **Tile Walk**

Format:	3D_Tilewalk

(This field must be set to 1: TILEWALK_YMAJOR)This field specifies the type of memory tiling (XMajor or YMajor) employed to tile this surface. See Memory Interface Functions for details on memory tiling and restrictions.This field is ignored when the surface is linear.This field is ignored by MFX. Internally H/W is always treated this set to 1 for all video codec and for JPEG.

Value	Name	Description
0h	XMAJOR	TILEWALK_XMAJOR

		1h YMAJOR TILEWALK_YMAJOR
		Programming Notes
		The corresponding cache(s) must be invalidated before a previously accessed surface is accessed again with an altered state of this bit
4	31	**Reserved**
		Format: MBZ
	30:16	**X Offset for U(Cb)**
		Project: All
		Format: U15 Pixel Offset
		This field specifies the horizontal offset in pixels from the Surface Base Address to the start (origin) of the U(Cb) plane or the interleaved UV plane if Interleave Chroma is enabled. This field is only used for PLANAR surface formats. This field must be set to zero.X Offset for U(Cb) in pixel (This field must be zero for NV12 and IMC 1 and 3)
		Programming Notes
		For PLANAR_420 and PLANAR_422 surface formats, this field must be zero.
	15	**Reserved**
		Project: All
		Format: MBZ
	14:0	**Y Offset for U(Cb)**
		Project: All
		Format: U15 Pixel Row Offset
		This field specifies the veritical offset in rows from the Surface Base Address to the start (origin) of the U(Cb) plane or the interleaved UV plane if Interleave Chroma is enabled. This field is only used for PLANAR surface formats.
		Programming Notes
		For PLANAR_420 and PLANAR_422 surface formats, this field must be multiple of 16 pixels – i.e. multiple MBs. For JPEG, it is block aligned
5	31:29	**Reserved**
		Format: MBZ
	28:16	**X Offset for V(Cr)**
		Format: U13 Offset in Pixels
		This field must be zero for NV12 and IMC 1 and 3
		This field specifies the horizontal offset in pixels from the Surface Base Address to the start (origin) of the V(Cr) plane. This field is only used for PLANAR surface formats with Interleave Chroma disabled.
		Programming Notes
		For PLANAR_420 and PLANAR_422 surface formats, this field must indicate an even number of pixels.
	15:0	**Y Offset for V(Cr)**

Format:	U16 Row Offset in Pixels

This field specifies the veritical offset in rows from the Surface Base Address to the start (origin) of the V(Cr) plane. This field is only used for PLANAR surface formats with Interleave Chroma disabled. This field is ignored by all video codec, only used by JPEG.

Programming Notes

For PLANAR_420 surface formats, this field must be multiple of 16 pixels – i.e. multiple MBs. For JPEG, it is block aligned

1.7.5 MFX_PIPE_BUF_ADDR_STATE Command

MFX_PIPE_BUF_ADDR_STATE

Source:	VideoCS
Length Bias:	2

This state command provides the memory base addresses for all row stores, StreamOut buffer and reconstructed picture output buffers required by the MFD or MFC Engine (that are in addition to the row stores of the Bit Stream Decoding/Encoding Unit (BSD/BSE) and the reference picture buffers). This is a picture level state command and is common among all codec standards and for both encoder and decoder operating modes. However, some fields may only applicable to a specific codec standard.All Pixel Surfaces (original, reference frame and reconstructed frame) in the Encoder are programmed with the same surface state (NV12 and TileY format), except each has its own frame buffer base address. In the tile format, there is no need to provide buffer offset for each slice; since from each MB address, the hardware can calculated the corresponding memory location within the frame buffer directly.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_PIPE_BUF_ADDR_STATE Format: OpCode
	26:24	**Common Opcode** Default Value: 0h MFX_PIPE_BUF_ADDR_STATE Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h MFX_PIPE_BUF_ADDR_STATE Format: OpCode
	20:16	**SubOpcode B** Default Value: 2h MFX_PIPE_BUF_ADDR_STATE Format: OpCode
	15:12	**Reserved** Project: All Format: MBZ
	11:0	**DWord Length** Project: All Format: =n Total Length Fixed Length Value: 16h — Name: DWORD_COUNT_n **[Default]** — Description: Excludes DWord (0,1)
1	31:6	**Pre Deblocking - Destination Address** Specifies the 4K byte aligned frame buffer address for outputting the non-filtered reconstructed YUV picture (i.e. output of final adder in each codec standard, and prior to the deblocking filter unit).This field is ignored if PreDeblockOutEnable is set to 0 (disable).

MFX_PIPE_BUF_ADDR_STATE

	5:0	**Reserved** Format: MBZ
2	31:6	**Post Deblocking - Destination Address** Specifies the 4K byte aligned frame buffer address for outputting the post-loop filtered reconstructed YUV picture (i.e. output of the deblocking filter unit)This field is ignored if PostDeblockOutEnable is set to 0 (disable).
	5:4	**Post Deblocking – Arbitration Priority Control** Format: U2 Enumerated Type This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.

Value	Name
00b	Highest priority
01b	Second highest priority
10b	Third highest priority
11b	Lowest priority

	2	**Post Deblocking - Graphics Data Type (GFDT)** Format: U1 This field contains the GFDT bit for this surface when writes occur. GFDT can also be set by the GTT. The effective GFDT is the logical OR of this field with the GFDT from the GTT entry. This field is ignored for reads.
	1:0	**Post Deblocking - Cacheability Control** Format: U2 Enumerated type This field controls cacheability in the mid-level cache (MLC) and last-level cache (LLC).

Value	Name	Description
00b	GTT entry	Use cacheability control bits from GTT entry
01b	not LLC or MLC	Data is not cached in LLC or MLC
10b	in LLC but not MLC	Data is cached in LLC but not MLC
11b	both LLC and MLC	Data is cached in both LLC and MLC

3	31:6	**Original Uncompressed Picture - Source Address (CurSrcAddr)** Exists If: Encoding Format: Address[31:6] Specifies the 64 byte aligned frame buffer address for fetching YUV pixel data from the original uncompressed input picture for encoding.
	5:4	**Original Uncompressed Picture – Arbitration Priority Control** Format: U2 Enyumerated Type This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.

Value	Name
00b	Highest priority
01b	Second highest priority
10b	Third highest priority
11b	Lowest priority

	2	**Original Uncompressed Picture - Graphics Data Type (GFDT)**

MFX_PIPE_BUF_ADDR_STATE

		Format: U1 This field contains the GFDT bit for this surface when writes occur. GFDT can also be set by the GTT. The effective GFDT is the logical OR of this field with the GFDT from the GTT entry. This field is ignored for reads.
	1:0	**Original Uncompressed Picture - Cacheability Control** Format: U2 Enumerated Type This field controls cacheability in the mid-level cache (MLC) and last-level cache (LLC). **Value** / **Name** / **Description** 00b / GTT entry / use cacheability control bits from GTT entry 01b / not in LLC or MLC / data is not cached in LLC or MLC 10b / in LLC but not MLC / data is cached in LLC but not MLC 11b / both LLC and MLC / data is cached in both LLC and MLC
4	31:6	**StreamOut Data Destination - Base Address (StreamOutAddr)** Format: StreamOutAddress[31:6] 64 byte aligned buffer Specifies the address for outputting the per-MB indirect data to memory when StreamOutEnable is set in the MFX_PIPE_MODE_SELECT command. For decoder : this field is used for transcoding purpose. For encoder : this field is used for dynamic repeat of frame in PAK for Rate Control. Also used for feeding coding information back to the Host, Video Preprocessing Unit and ENC Unit.All data are written in fixed formats, and therefore all record sizes are known in the hardware. Hardware can calculate the offset into this base address for per-MB data.
	5:4	**StreamOut Data Destination – Arbitration Priority Control** Project: All Format: U2 Enumerated Type This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. **Value** / **Name** 00b / Highest priority 01b / Second highest priority 10b / Third highest priority 11b / Lowest priority
	2	**StreamOut Data Destination - Graphics Data Type (GFDT)** Format: U1 This field contains the GFDT bit for this surface when writes occur. GFDT can also be set by the GTT. The effective GFDT is the logical OR of this field with the GFDT from the GTT entry. This field is ignored for reads.
	1:0	**StreamOut Data Destination - Cacheability Control** Format: U2 Enumerated Type This field controls cacheability in the mid-level cache (MLC) and last-level cache (LLC). **Value** / **Name** / **Description**

		00b	GTT entry	use cacheability control bits from GTT entry
		01b	Not in LLC or MLC	data is not cached in LLC or MLC
		10b	In LLC but not MLC	data is cached in LLC but not MLC
		11b	Both LLC and MLC	data is cached in both LLC and MLC

5	31:6	**Intra Row Store Scratch Buffer - Base Address (IntraOSRowStoreAddr)** Format: GraphicsAddress[31:6] This field provides the base address of the scratch buffer (read/write) used by the AVC IntraPrediction unit to store MB information of the previous row for processing of each macroblock in the current row. The Intra Row Store buffer must be 64-byte cacheline aligned.Hardware uses the horizontal address of the current macroblock to address the Intra Row Store.This field is ignored in MPEG2 and VC1 mode.Max 256 cachelines for 4K pixels (1 cacheline for either MBAFF or non-MBAFF)
	5:4	**Intra/Overlap Smoothing Row Store Scratch Buffer – Arbitration Priority Control** Format: U2 Enumerated Type This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.

Value	Name
00b	Highest priority
01b	Second highest priority
10b	Third highest priority
11b	Lowest priority

	2	**Intra/Overlap Smoothing Row Store Scratch Buffer - Graphics Data Type (GFDT)** Format: U1 This field contains the GFDT bit for this surface when writes occur. GFDT can also be set by the GTT. The effective GFDT is the logical OR of this field with the GFDT from the GTT entry. This field is ignored for reads.
	1:0	**Intra/Overlap Smoothing Row Store Scratch Buffer - Cacheability Control** Format: U2 Enumerated Type This field controls cacheability in the mid-level cache (MLC) and last-level cache (LLC).

Value	Name	Description
00b	GTT entry	use cacheability control bits from GTT entry
01b	Not in LLC or MLC	data is not cached in LLC or MLC
10b	In LLC but not MLC	data is cached in LLC but not MLC
11b	Both LLC and MLC	data is cached in both LLC and MLC

6	31:6	**Deblocking Filter Row Store Scratch Buffer - Base Address (DeblockRowStoreAddr** Format: GraphicsAddress[31:6] Deblocking Filter Row Store is needed for VC1 Overlap-smoothing Filter This field provides the base address of the scratch buffer (read and write) used by the deblocking filter unit to store MB information of the previous row for filtering of each macroblock in the current row. The Deblocking Filter Row Store buffer must be 64-byte cacheline aligned. Hardware uses the horizontal address of the current macroblock to address the Deblocking Filter Row Store. Max 6 cachelines for VC1 and MPEG2, and max 4 for AVC (for MBAFF, 2 for non-MBAFF).

MFX_PIPE_BUF_ADDR_STATE

	5:4	**Deblocking Filter Row Store Scratch Buffer – Arbitration Priority Control** Format: U2 Enumerated Type This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. **Value** / **Name** 0h – Highest priority 1h – Second highest priority 1h – Third highest priority 1h – Lowest priority
	2	**Deblocking Filter Row Store Scratch Buffer - Graphics Data Type (GFDT)** Project: All Format: U1 This field contains the GFDT bit for this surface when writes occur. GFDT can also be set by the GTT. The effective GFDT is the logical OR of this field with the GFDT from the GTT entry. This field is ignored for reads.
	1:0	**Deblocking Filter Row Store Scratch Buffer - Cacheability Control** Project: All Format: U2 Enumerated Type This field controls cacheability in the mid-level cache (MLC) and last-level cache (LLC). **Value** / **Name** / **Description** 00b – GTT entry – use cacheability control bits from GTT entry 01b – Not in LLC or MLC – data is not cached in LLC or MLC 10b – LLC but not MLC – data is cached in LLC but not MLC 11b – both LLC and MLC – data is cached in both LLC and MLC
7..22	31:6	**Reference Picture (RefAddr[0-15]) - Addresses** Format: GraphicsAddress[31:6] Specifies the 64 byte aligned reference frame buffer addresses for the motion compensation operation in AVC/VC1/MPEG2. AVC can specify up to 16 YUV frame-based surfaces for both forward and backward references, i.e. L0+L1 total = 16 max. Any entry can be assigned to L0 or L1 or both lists.But VC1 and MPEG2, worst case, can use up to 2 YUV frame-based surfaces for both forward and backward references:P-MB : RefAddr[0] – temporal closest previous field of a reference frame (can be the current frame)RefAddr[1] – next temporal closest previous field of a reference frame (must be different from the current frame)It is a variant (without the LongTermRefPic specification) of the RefFrameList[16] defined in AVC DXVA Spec. RefAddr[0-15] is indexed by frame_storeID >>1. It is not a packed list, i.e. invalid entries can scatter among the list. All invalid addresses must be set to a valid address RefAddr[0] by the driver. The same applies to VC1 and MPEG2. **Programming Notes** AVC: Always specifies all 16 addresses even some of them are not needed as indicated by the max num of active reference pictures. This is done for preventing data corruption (error, fault condition, etc.) by having all the references being set to a legal location.
	5:4	**Reference Picture (RefAddr[0-15]) – Arbitration Priority Control** Format: U2 Enumerated Type This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. **Value** / **Name**

00b	Highest priority	
01b	Second highest priority	
10b	Third highest priority	
11b	Lowest priority	

2 **Reference Picture (RefAddr[0-15]) - Graphics Data Type (GFDT)**

Project:	All
Format:	U1

This field contains the GFDT bit for this surface when writes occur. GFDT can also be set by the GTT. The effective GFDT is the logical OR of this field with the GFDT from the GTT entry. This field is ignored for reads.H/W only reads this bit from the very first RefAddr[0][bit 3:0], all other RefAddr[i][bit 3:0] are ignored by H/W and are assumed to have the same values as that of RefAddr[0].

1:0 **Reference Picture (RefAddr[0-15]) - Cacheability Control**

Project:	All
Format:	U2 Enumerated Type

This field controls cacheability in the mid-level cache (MLC) and last-level cache (LLC).H/W only reads this bit from the very first RefAddr[0][bit 3:0], all other RefAddr[i][bit 3:0] are ignored by H/W and are assumed to have the same values as that of RefAddr[0].

Value	Name	Description
00b	GTT entry	use cacheability control bits from GTT entry
01b	not in LLC or MLC	data is not cached in LLC or MLC
10b	in LLC but not MLC	data is cached in LLC but not MLC
11b	both LLC and MLC	data is cached in both LLC and MLC

23 31:6 **Macroblock Status Buffer Base Address (MacroblockStatAddr)**

Project:	All
Format:	MacroblockStatusAddress[31:6] 64 byte aligned buffer

Specifies the address for reading the per-MB indirect data from memory when MacroblockStatEnable is set in the MFX_AVC_IMG_STATE Command.For decoder : this field is ignored by hardware.For encoder: this field is used for dynamic repeat of frame in PAK for Rate Control. Also used for feeding coding information back to the Host, Video Preprocessing Unit and ENC Unit.All data are written in fixed formats, and therefore all record sizes are known in the hardware. Hardware can calculate the offset into this base address for per-MB data.

5:4 **Arbitration Priority Control**

Project:	All
Format:	U2 Enumerated Type

This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.

Value	Name
00b	Highest priority
01b	Second highest priority
10b	Third highest priority
11b	Lowest priority

2 **Graphics Data Type (GFDT**

Project:	All

MFX_PIPE_BUF_ADDR_STATE

		Format: U1 This field contains the GFDT bit for this surface when writes occur. GFDT can also be set by the GTT. The effective GFDT is the logical OR of this field with the GFDT from the GTT entry. This field is ignored for reads.
	1:0	**Cacheability Control** Project: All Format: U2 Enumerated Type This field controls cacheability in the mid-level cache (MLC) and last-level cache (LLC). **Value** / **Name** / **Description** 00b / GTT / use cacheability control bits from GTT entry 01b / Not in LLC or MLC / data is not cached in LLC or MLC 10b / In LLC but not MLC / data is cached in LLC but not MLC 11b / both LLC and MLC / data is cached in both LLC and MLC
24	31:1	**Reserved** Format: MBZ
	0	**Reserved** Format: MBZ

1.7.6 MFX_IND_OBJ_BASE_ADDR_STATE Command

MFX_IND_OBJ_BASE_ADDR_STATE	
Source:	VideoCS
Length Bias:	2

This state command provides the memory base addresses for all row stores, StreamOut buffer and reconstructed picture output buffers required by the MFD or MFC Engine (that are in addition to the row stores of the Bit Stream Decoding/Encoding Unit (BSD/BSE) and the reference picture buffers). This is a picture level state command and is common among all codec standards and for both encoder and decoder operating modes. However, some fields may only applicable to a specific codec standard. All Pixel Surfaces (original, reference frame and reconstructed frame) in the Encoder are programmed with the same surface state (NV12 and TileY format), except each has its own frame buffer base address. In the tile format, there is no need to provide buffer offset for each slice; since from each MB address, the hardware can calculated the corresponding memory location within the frame buffer directly.

The MFX_IND_OBJ_BASE_ADDR command sets the memory base address pointers for the corresponding Indirect Object Data Start Addresses (Offsets) specified in each OBJECT commands. The characteristic of these indirect object data is their variable size (per MB or per Slice). Hence, each OBJECT command must specify the indirect object data offset from the base address to start fetching or writing object data. While the use of base address is unconditional, the indirection can be effectively disabled by setting the base address to zero. For decoder, there are only 1 read-only per-slice indirect object in the BSD_OBJECT Command, and2 read-only per-MB indirect objects in the IT_OBJECT CommandFor decoder : the Video Command Streamer (VCS) will perform the memory access bound check automatically using the corresponding MFC Indirect Object Access Upper Bound specification. If any access is at or beyond the upper bound, zero value is returned. The request to memory is still being sent, but the corresponding codec's BSD unit will detect this condition and perform the zeroing return. If the Upper Bound is turned off, the beyond bound request will return whatever on the bus (invalid data).For encoder, there are 1 read-only per-MB indirect object in the PAK_OBJECT Command, and1 write-only per-slice indirect object in the PAK Slice_State CommandFor encoder : whenever an out of bound address accessing request is generated, VMX will detect such requests and snap the address to the corresponding [indirect object base address + indirect data start address]. VMX will return all 0s as the data to the requestor. NotationDefinitionPhysicalAddress[n:m]Corresponding bits of a physical graphics memory byte address (not mapped by a GTT)GraphicsAddress[n:m]Corresponding bits of an absolute, virtual graphics memory byte address (mapped by a GTT).

DWord	Bit	Description	
0	31:29	**Command Type**	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	**Pipeline**	
		Default Value:	2h MFX_IND_OBJ_BASE_ADDR_STATE
		Format:	OpCode
	26:24	**Common Opcode**	
		Default Value:	0h MFX_IND_OBJ_BASE_ADDR_STATE
		Format:	OpCode
	23:21	**Sub OpcodeA**	
		Default Value:	0h MFX_IND_OBJ_BASE_ADDR_STATE
		Format:	OpCode
	20:16	**SubOpcodeB**	
		Default Value:	3h MFX_IND_OBJ_BASE_ADDR_STATE
		Format:	OpCode
	15:12	**Reserved**	
		Project:	All

MFX_IND_OBJ_BASE_ADDR_STATE

		Format: MBZ
	11:0	**DWord Length** Default Value: 0009h Excludes DWord (0,1) Project: All Format: =n Total Length - 2
1	31:12	**MFX Indirect Bitstream Object - Base Address (Decoder and Stitch Modes)** Project: All Format: GraphicsAddress[31:12] Specifies the 4K-byte aligned memory base address for the read-only indirect data object pointed in the MFD_XXX_BSD_OBJECT command for fetching (reading) the compressed Slice Data.This field is only valid in MPEG2, AVC and VC1 decoder VLD mode.
	11:6	**Reserved** Project: All Format: MBZ
	5:4	**MFX Indirect BSD Object – Arbitration Priority Control** Project: All Format: U2 Enumerated Type This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. Value / Name: 00b Highest priority 01b Second highest priority 10b Third highest priority 11b Lowest priority
	2	**MFX Indirect Bitstream Object - Graphics Data Type (GFDT)** Project: All Format: U1 This field contains the GFDT bit for this surface when writes occur. GFDT can also be set by the GTT. The effective GFDT is the logical OR of this field with the GFDT from the GTT entry. This field is ignored for reads.
	1:0	**MFX Indirect Bitstream Object - Cacheability Control** Project: All Format: U2 Enumerated Type This field controls cacheability in the mid-level cache (MLC) and last-level cache (LLC). Value / Name / Description: 00b GTT entry use cacheability control bits from GTT entry 01b not in LLC or MLC data is not cached in LLC or MLC 10b in LLC but not MLC data is cached in LLC but not MLC 11b both LLC and MLC data is cached in both LLC and MLC
2	31:12	**MFX Indirect Bitstream Object - Access Upper Bound (Decoder and Stitch Modes)** Project: All Format: GraphicsAddress[31:12]

MFX_IND_OBJ_BASE_ADDR_STATE

This field specifies the 4K-byte aligned (exclusive) maximum Graphics Memory address access by the indirect data object in the MFD_XXX_BSD_OBJECT command for the compressed Slice Data. Indirect data accessed at this address and beyond will return as 0 by the hardware. Setting this field to 0 will cause this range check to be ignored.If non-zero, this address must be greater than the MFX Indirect Bitstream ObjectBase Address state.Hardware ignores this field if indirect data is not present, i.e. the Indirect Data Length field of the MFD_XXX_BSD_OBJECT command is set to 0.This field is only valid in MPEG2, AVC and VC1 decoder VLD mode.

11:0 Reserved

Project:	All
Format:	MBZ

3 — 31:12 MFX Indirect MV Object - Base Address

Project:	All
Format:	GraphicsAddress[31:12]

Specifies the 4K-byte aligned memory base address for the read-only indirect data object pointed in the encoder MFC_AVC_PAK_OBJECT command or the decoder MFD_IT_OBJECT command for fetching the per-MB MV data.This field is only valid in AVC encoder mode or in AVC decoder IT mode

11:6 Reserved

Project:	All
Format:	MBZ

5:4 MFX Indirect MV Object - Arbitration Priority Control

Project:	All
Format:	U2 Enumerated Type

This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.

Value	Name
00b	Highest priority
01b	Second highest priority
10b	Third highest priority
11b	Lowest priority

2 MFX Indirect MV Object - Graphics Data Type (GFDT)

Project:	All
Format:	U1

This field contains the GFDT bit for this surface when writes occur. GFDT can also be set by the GTT. The effective GFDT is the logical OR of this field with the GFDT from the GTT entry. This field is ignored for reads.

1:0 MFX Indirect MV Object - Cacheability Control

Project:	All
Format:	U2

This field controls cacheability in the mid-level cache (MLC) and last-level cache (LLC).

Value	Name	Description
00b	From GTT entry	use cacheability control bits from GTT entry
01b	Not cached in LLC or MLC	data is not cached in LLC or MLC
10b	In LLC but not MLC	data is cached in LLC but not MLC
11b	Both LLC and MLC	data is cached in both LLC and MLC

4 — 31:12 MFX Indirect MV Object Access Upper Bound

Project:	All

MFX_IND_OBJ_BASE_ADDR_STATE

		Format: GraphicsAddress[31:12] This field specifies the 4K-byte aligned (exclusive) maximum Graphics Memory address access by the indirect data object in the MFC_AVC_PAK_OBJECT / MFD_IT_OBJECT command for the per-MB MV data. Indirect data accessed at this address and beyond will return as 0 by the hardware. Setting this field to 0 will cause this range check to be ignored.If non-zero, this address must be greater than the MFX Indirect MV Object Base Address state.Hardware ignores this field if indirect data is not present, i.e. the Indirect Data Length field of the MFC_AVC_PAK_OBJECT / MFD_IT_OBJECT command is set to 0.This field is only valid in AVC encoder mode or in AVC decoder IT mode.
	11:0	**Reserved** Project: All Format: MBZ
5	31:12	**MFD Indirect IT-COEFF Object - Base Address (Decoder Only)** Project: All Format: GraphicsAddress[31:12] Specifies the 4K-byte aligned memory base address for the read-only indirect data object pointed in the MFD_IT_OBJECT command for fetching (reading) the per-MB non-scaled coefficient data (all inverse scaling and quantization are done in hardware).This field is only valid in MPEG2, AVC and VC1 decoder IT mode.
	11:6	**Reserved** Project: All Format: MBZ
	5:4	**MFD Indirect IT-COEFF Object - Arbitration Priority Control** Project: All Format: U2 Enumerated Type This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. Value / Name: 00b Highest priority 01b Second highest priority 10b Third highest priority 11b Lowest priority
	2	**MFD Indirect IT-COEFF Object - Graphics Data Type (GFDT)** Project: All Format: U1 This field contains the GFDT bit for this surface when writes occur. GFDT can also be set by the GTT. The effective GFDT is the logical OR of this field with the GFDT from the GTT entry. This field is ignored for reads.
	1:0	**MFD Indirect IT-COEFF Object - Cacheability Control** Project: All Format: U2 Enumerated type This field controls cacheability in the mid-level cache (MLC) and last-level cache (LLC). Value / Name / Description: 00b From GTT entry — use cacheability control bits from GTT entry 01b Not in LLC or MLC — data is not cached in LLC or MLC

MFX_IND_OBJ_BASE_ADDR_STATE

		10b	In LLC but not MLC	data is cached in LLC but not MLC
		11b	Both LLC and MLC	data is cached in both LLC and MLC

6	31:12	**MFD Indirect IT-COEFF Object - Access Upper Bound (Decoder Only)**

Project:	All
Format:	GraphicsAddress[31:12]

This field specifies the 4K-byte aligned (exclusive) maximum Graphics Memory address access by the indirect data object in the MFD_IT_OBJECT command for the per-MB non-scaled coefficient data. Indirect data accessed at this address and beyond will return as 0 by the hardware. Setting this field to 0 will cause this range check to be ignored.If non-zero, this address must be greater than the MFD Indirect IT-COEFF Object Base Address state.Hardware ignores this field if indirect data is not present, i.e. the Indirect COEFF Data Length field of the MFD_IT_OBJECT command is set to 0.This field is only valid in MPEG2, AVC and VC1 decoder IT mode.

	11:0	**Reserved**

Project:	All
Format:	MBZ

7	31:12	**MFD Indirect IT-DBLK Object - Base Address (Decoder Only)**

Project:	All
Format:	GraphicsAddress[31:12]

Specifies the 4K-byte aligned memory base address for the read-only indirect data object pointed in the MFD_IT_OBJECT command for fetching (reading) the per-MB Deblocking filter control data.This field is only valid in AVC decoder IT mode.

	11:6	**Reserved**

Project:	All
Format:	MBZ

	5:4	**MFD Indirect IT-DBLK Object - Arbitration Priority Control**

Project:	All
Format:	U2 Enumerated Type

This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.

Value	Name
00b	Highest priority
01b	Second highest priority
10b	Third highest priority
11b	Lowest priority

	2	**MFD Indirect IT-DBLK Object - Graphics Data Type (GFDT)**

Project:	All
Format:	U1

This field contains the GFDT bit for this surface when writes occur. GFDT can also be set by the GTT. The effective GFDT is the logical OR of this field with the GFDT from the GTT entry. This field is ignored for reads.

	1:0	**MFD Indirect IT-DBLK Object - Cacheability Control**

Project:	All
Format:	U2 Enumerated Type

MFX_IND_OBJ_BASE_ADDR_STATE

This field controls cacheability in the mid-level cache (MLC) and last-level cache (LLC).

Value	Name	Description	Project
00b	From GTT entry	use cacheability control bits from GTT entry	All
01b	Not cached in LLC or MLC	data is not cached in LLC or MLC	All
10b	In LLC but not MLC	data is cached in LLC but not MLC	All
11b	Both LLC and MLC	data is cached in both LLC and MLC	All

8 — 31:12 — MFD Indirect IT-DBLK Object Access Upper Bound (Decoder Only)

Project:	All
Format:	GraphicsAddress[31:12]
Format:	GraphicsAddress[31:12]

This field specifies the 4K-byte aligned (exclusive) maximum Graphics Memory address access by the indirect data object in the MFD_IT_OBJECT command for the per-MB Deblocking filter control data. Indirect data accessed at this address and beyond will return as 0 by the hardware. Setting this field to 0 will cause this range check to be ignored.If non-zero, this address must be greater than the MFD Indirect IT-DBLK Object Base Address state.Hardware ignores this field if indirect data is not present, i.e. the Indirect Deblocking Control Data Length field of the MFD_IT_OBJECT command is set to 0.This field is only valid in AVC decoder IT mode.

11:0 — Reserved

Project:	All
Format:	MBZ

9 — 31:12 — MFC Indirect PAK-BSE Object - Base Address (Encoder Only)

Project:	All
Format:	GraphicsAddress[31:12]

Specifies the 4K-byte aligned memory base address for the write-only indirect data object pointed in the PAK_SLICE_STATE command for writing out the compressed bitstream.This field is only valid in AVC encoder mode.

11:6 — Reserved

Project:	All
Format:	MBZ

5:4 — MFC Indirect PAK-BSE Object - Arbitration Priority Control

Project:	All
Format:	U2 Enumerated Type

This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.

Value	Name
00b	Highest priority
01b	Second highest priority
10b	Third highest priority
11b	Lowest priority

2 — MFC Indirect PAK-BSE Object - Graphics Data Type (GFDT)

Project:	All
Format:	U1

This field contains the GFDT bit for this surface when writes occur. GFDT can also be set by the GTT. The effective GFDT is the logical OR of this field with the GFDT from the GTT entry. This field is ignored for reads.

MFX_IND_OBJ_BASE_ADDR_STATE

	1:0	**MFC Indirect PAK-BSE Object - Cacheability Control** Project: All Format: U2 Enumerated Type This field controls cacheability in the mid-level cache (MLC) and last-level cache (LLC). **Value** / **Name** / **Description** 00b / GTT entry / use cacheability control bits from GTT entry 01b / Not in LLC or MLC / data is not cached in LLC or MLC 10b / In LLC but not MLC / data is cached in LLC but not MLC 11b / Both LLC and MLC / data is cached in both LLC and MLC
10	31:12	**MFC Indirect PAK-BSE Object - Access Upper Bound (Encoder Only)** Project: All Format: GraphicsAddress[31:12] This field specifies the 4K-byte aligned (exclusive) maximum Graphics Memory address access by the indirect data object in the PAK_SLICE_STATE command for the per-slice output bitstream. Indirect data accessed at this address and beyond will be blocked by the hardware and ignored. Setting this field to 0 will cause this range check to be ignoredIf non-zero, this address must be greater than the MFC Indirect PAK-BSE Object Base Address state.This field is only valid in AVC encoder mode.
	11:0	**Reserved** Project: All Format: MBZ

1.7.7 MFX_PAK_INSERT_OBJECT

MFX_PAK_INSERT_OBJECT

Source: VideoCS

Length Bias: 2

Description	Project
The MFX_PAK_INSERT_OBJECT command is the first primitive command for the AVC and MPEG2 Encoding Pipeline.	
This command is issued to setup the control and parameters of inserting a chunk of compressed/encoded bits into the current bitstream output buffer starting at the specified bit locationto perform the actual insertion by transferring the command inline data to the output buffer max, 32 bits at a time. It is a variable length command as the data to be inserted are presented as inline data of this command. It is a multiple of 32-bit (1 DW), as the data bus to the bitstream buffer is 32-bit wide. Multiple insertion commands can be issued back to back in a series. It is host software's responsibility to make sure their corresponding data will properly stitch together to form a valid H.264 bitstream. Internally, MFX hardware will keep track of the very last two bytes' (the very last byte can be a partial byte) values of the previous insertion. It is required that the next Insertion Object Command or the next PAK Object Command to perform the start code emulation sequence check and prevention 0x03 byte insertion with this end condition of the previous insertion.	

MFX_PAK_INSERT_OBJECT

Hardware will keep track of an output bitstream buffer current byte position and the associated next bit insertion position index. Data to be inserted can be a valid H.264 NAL units or a partial NAL unit. Certain NAL unit has a minimum byte size requirement. As such the hardware will optionally (enabled by STATE Command) determines the number of CABAC_ZERO_WORD to be inserted to the end of the current NAL, based on the minimum byte size of a NAL and the actual bin count of the encoded Slice. Since prior to the CABAC_ZERO_WORD insertion, the RBSP or EBSP is already byte-aligned, so each CABAC_ZERO_WORD insertion is actually a 3-byte sequence 0x00 00 03. The inline data may have already been processed for start code emulation byte insertion, except the possibility of the last 2 bytes plus the very last partial byte (if any). Hence, when hardware performing the concatenation of multiple consecutive insertion commands, or concatenation of an insertion command and a PAK object command, it must check and perform the necessary start code emulation byte insert at the junction.The inline data is required to be byte aligned on the left (first transmitted bit order) and may or may not be byte aligned on the right (last transmitted bits).

The command will specify the bit offset of the last valid DW.Each insertion state command defines a chunk of bits (compressed data) to be inserted at a specific location of the output compressed bitstream in the output buffer.Depend on CABAC or CAVLC encoding mode (from Slice State), PAK Object Command is always ended in byte aligned output bitstream except for CABAC header insertion which is bit aligned. In the aligned cases, PAK will perform 0 filling in CAVLC mode, and 1 filling in CABAC mode.

Insertion data can include:any encoded syntax elements bit data before the encoded Slice Data (PAK Object Command) of the current SliceSPS NALPPS NALSEI NALOther Non-Slice NALLeading_Zero_8_bits (as many bytes as there is)Start Code PrefixNAL Header ByteSlice HeaderAny encoded syntax elements bit data after the encoded Slice Data (PAK Object Command) of the current Slice and prior to the next encoded Slice Data of the next Slice or prior to the end of the bistream, whichever comes firstCabac_Zero_Word or Trailing_Zero_8bits (as many bytes as there is).

Anything listed above before a Slice DataContext switch interrupt is not supported by this command.

DWord	Bit	Description	
0	31:29	**Command Type**	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	**Pipeline**	
		Default Value:	2h MFX_PAK_INSERT_OBJECT
		Format:	OpCode
	26:24	**Media Command Opcode**	
		Default Value:	0h MFX_COMMON
		Format:	OpCode
	23:21	**SubOpcode A**	
		Default Value:	2h
		Format:	OpCode
	20:16	**SubOpcode B**	
		Default Value:	8h
		Format:	OpCode
	15:12	**Reserved**	
		Project:	All
		Format:	MBZ
	11:0	**DWord Length**	
		Default Value:	0h Excludes DWord (0,1) = Variable Length in DW
		Project:	All

MFX_PAK_INSERT_OBJECT

		Format: =n Total Length - 2
1	31:18	**Reserved** Format: MBZ
	17:16	**DataByteOffset – SrcDataStartingByteOffset[1:0]** Source Data Starting Byte Position within the very first inline DW.
	15:14	**Reserved** Format: MBZ
	13:8	**DataBitsInLastDW – SrCDataEndingBitInclusion[5:0]** Source Data to be included in the very last inline DW. Follows the MSBit is the upper bit of each byte within the DW. The lower byte is actually processed first.For example, SrCDataEndingBitInclusion = 9, bit 7:0 and bit 15 are included as valid header data. **Value** / **Name** [1,32]
	7:4	**SkipEmulByteCnt – Skip Emulation Byte Count** Skip emulation check for number of starting bytesIt can be programmed from 0 to 15 bytes.For example, to skip the start code that has already prefixed in the bitstream.
	3	**EmulationFlag – EmulationByteBitsInsertEnable** **Value** / **Name** 1 instruct the hardware to perform Start Code Prefix (0x 00 00 01/02/03/00) Search and Prevention Byte (0x 03) insertion on the insertion data of this command. It is required that hardware will handle a start code prefix crossing the boundary between 2 insertion commands, or an insertion command followed by a PAK Object command.
	2	**LastHeaderFlag – LastSrcHeaderDataInsertCommandFlag** To process a series of consecutive insertion commands, this flag (=1) indicates the current command is the last 'header' insertion in the series.In CABAC, hardware must perform the "1" insert for byte align for Slice Header before Slice Data comes in in the next PAK-OBJECT command.In CAVLC, hardware ignores this bit
	1	**EndOfSliceFlag – LastDstDataInsertCommandFlag** No more insertion command and no more PAK-OBJECT command follows.Flush data out to memory
	0	**BitstreamStartReset – ResetBitStreamStartingPos** **Value** / **Name** 1 Reset the bitstream buffer insertion position to the bitstream buffer starting position. 0 Insert the current command inline data starting at the current bitstream buffer insertion position
2..n	31:0	**Insert Data PayLoad** Actual Data to be inserted to the output bitstream buffer.

1.7.8 MFX_STITCH_OBJECT

MFX_STITCH_OBJECT	
Project:	All
Source:	VideoCS
Length Bias:	2

The MFC_STITCH_OBJECT command is used when stitch-enabled is set to 1, while CodecSel and StandardSel are set to ENCODE and AVC, respectively.This command is used, for example, to stitch multiple bitstreams to form a transport stream. . It is a variable length command as the data to be inserted are presented as either inline data and/or indirect data of this command. Multiple insertion commands can be issued back to back in a series. It is host software's responsibility to make sure their corresponding data will properly stitch together to form a valid output. Hardware keeps track of an output bitstream buffer current byte position and the associated next bit insertion position index. Context switch interrupt is not supported by this command. In order to support interrupt, it is software's responsibility to set up ARB_ON/OFF commands at the proper position to allow interrupt.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFC_STITCH_OBJECT Format: OpCode
	26:24	**Media Command Opcode** Default Value: 0h MFX_COMMON Format: OpCode
	23:21	**SubOpcode A** Default Value: 2h Format: OpCode
	20:16	**SubOpcode B** Default Value: Ah Format: OpCode
	15:12	**Reserved** Project: All Format: MBZ
	11:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) = Variable Length in DW (>= 3) Format: =n Total Length - 2 If it is 3, it indicates the absent of inline data.
1	31:18	**Reserved** Format: MBZ
	17:16	**Source Data Starting Byte Offset** Source Data Starting Byte Position within the very first inline DW.
	15:14	**Reserved** Format: MBZ
	13:8	**Source Data Ending Bit Inclusion** Source Data to be included in the very last inline DW. Follows the MSBit is the upper bit of each byte

		within the DW. The lower byte is actually processed first.For example, SrCDataEndingBitInclusion =9, bit 7:0 and bit 15 are included as valid header data. Value: [1,32] — Name:
	2	**Last Source Header Data Insert Command Flag** To process a series of consecutive insertion commands, this flag (=1) indicates the current command is the last 'header' insertion in the series.In CABAC, hardware must perform the "1" insert for byte align for Slice Header before Slice Data comes in in the next PAK-OBJECT command.In CAVLC, hardware ignores this bit.
	1	**Last Destination Data Insert Command Flag** THIS FIELD MUST BE THE SAME AS Last Source Header Data Insert Command Flag No more insertion command and no more PAK-OBJECT command follows.Flush data out to memory
2	31:19	**Reserved** Project: All Format: MBZ
	18:0	**Indirect Data Length** Format: U19 This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled – subsequently, the Indirect Data Start Address field is ignored. This field must have the same alignment as the Indirect Object Data Start Address.
3	31:0	**Indirect Data Start Address** Format: MfxIndirectBitstreamObjectAddress[31:0] This field specifies the Graphics Memory starting address of the data to be loaded into the kernel for processing. This pointer is relative to the MFX Indirect Bitstream Object Base Address. Hardware ignores this field if indirect data is not present. Value: [0,FFFFFFFFh] — Name:
4..n	31:0	**Insert Data PayLoad** Inline data to be inserted to the output bitstream buffer

1.7.9 MFX_QM_STATE Command

MFX_QM_STATE			
Source:		VideoCS	
Length Bias:		2	
This is a common state command for AVC encoder modes. For encoder, it represents both the forward QM matrices as well as the decoding QM matrices.This is a Frame-level state. Only Scaling Lists specified by an application are being sent to the hardware. The driver is responsible for determining the final set of scaling lists to be used for decoding the current slice, based on the AVC Spec Table 7-2 (Fall-Back Rules A and B).In MFX AVC PAK mode, PAK needs both forward Q scaling lists and IQ scaling lists. The IQ scaling lists are sent as in MFD in raster scan order. But the Forward Q scaling lists are sent in column-wise raster order (column-by-column) to simplify the H/W. Driver will perform all the scan order conversion for both ForwardQ and IQ.			
DWord	**Bit**	**Description**	
0	31:29	**Command Type**	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	**Pipeline**	
		Default Value:	2h MFX_MULTI_DW
		Format:	OpCode
	26:24	**Media Command Opcode**	
		Default Value:	0h MFX_COMMON_STATE
		Format:	OpCode
	23:21	**SubOpcode A**	
		Default Value:	0h
		Format:	OpCode
	20:16	**SubOpcode B**	
		Default Value:	8h
		Format:	OpCode
	15:12	**Reserved**	
		Project:	All
		Format:	MBZ
	11:0	**DWord Length**	
		Default Value:	20h Excludes DWord (0,1)
		Project:	All
		Format:	=n Total Length - 2
1	31:2	**Reserved**	
		Format:	MBZ
	1:0	**AVC or MPEG2 or JPEG**	
		For AVC QM Type: This field specifies which Quantizer Matrix is loaded.	
		For MPEG2 QM Type: This field specifies which Quantizer Matrix is loaded.	
		Value	**Name** / **Exists If**
		0	AVC_4x4_Intra_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs) — AVC- Decoder Only
		1	AVC_4x4_Inter_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved- — AVC- Decoder Only

MFX_QM_STATE

			4DWs)	
		2	AVC_8x8_Intra_MATRIX	AVC- Decoder Only
		3	AVC_8x8_Inter_MATRIX	AVC- Decoder Only
		0	MPEG_INTRA_QUANTIZER_MATRIX	MPEG2- Decoder Only
		1	MPEG_NON_INTRA_QUANTIZER_MATRIX	MPEG2- Decoder Only
		2-3	Reserved	MPEG2- Decoder Only
		Programming Notes		
		For JPEG encoder, each quantization element presents 16-bit 1/QM[i][j].		
2..33	31:0	**Forward Quantizer Matrix**		
		Project:	All	
		Format:	U32	
		The format of a Quantizer Matrix is an 8x8 matrix in raster order. Each element is an unsigned byte.		

Bits	31:24	23:16	15:8	7:0
Dword 1	QuantMatrix[0][3]	QuantMatrix[0][2]	QuantMatrix[0][1]	QuantMatrix[0][0]
Dword 2	QuantMatrix[0][7]	QuantMatrix[0][6]	QuantMatrix[0][5]	QuantMatrix[0][4]
Dword 3	QuantMatrix[1][3]	QuantMatrix[1][2]	QuantMatrix[1][1]	QuantMatrix[1][0]
…	…	…	…	…
Dword 16	QuantMatrix[7][7]	QuantMatrix[7][6]	QuantMatrix[7][5]	QuantMatrix[7][4]

1.7.10 MFX_FQM_STATE Command

MFX_QM_STATE

Source:	VideoCS
Length Bias:	2

This is a common state command for AVC encoder modes. For encoder, it represents both the forward QM matrices as well as the decoding QM matrices.This is a Frame-level state. Only Scaling Lists specified by an application are being sent to the hardware. The driver is responsible for determining the final set of scaling lists to be used for decoding the current slice, based on the AVC Spec Table 7-2 (Fall-Back Rules A and B).In MFX AVC PAK mode, PAK needs both forward Q scaling lists and IQ scaling lists. The IQ scaling lists are sent as in MFD in raster scan order. But the Forward Q scaling lists are sent in column-wise raster order (column-by-column) to simplify the H/W. Driver will perform all the scan order conversion for both ForwardQ and IQ.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_MULTI_DW Format: OpCode
	26:24	**Media Command Opcode** Default Value: 0h MFX_COMMON_STATE Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 8h Format: OpCode
	15:12	**Reserved** Project: All Format: MBZ
	11:0	**DWord Length** Default Value: 20h Excludes DWord (0,1) Project: All Format: =n Total Length - 2
1	31:2	**Reserved** Format: MBZ
	1:0	**AVC or MPEG2 or JPEG** **For AVC QM Type**: This field specifies which Quantizer Matrix is loaded. **For MPEG2 QM Type**: This field specifies which Quantizer Matrix is loaded.

Value	Name	Exists If
0	AVC_4x4_Intra_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs)	AVC- Decoder Only

MFX_QM_STATE

		1	AVC_4x4_Inter_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs)	AVC- Decoder Only
		2	AVC_8x8_Intra_MATRIX	AVC- Decoder Only
		3	AVC_8x8_Inter_MATRIX	AVC- Decoder Only
		0	MPEG_INTRA_QUANTIZER_MATRIX	MPEG2- Decoder Only
		1	MPEG_NON_INTRA_QUANTIZER_MATRIX	MPEG2- Decoder Only
		2-3	Reserved	MPEG2- Decoder Only
		Programming Notes		
		For JPEG encoder, each quantization element presents 16-bit 1/QM[i][j].		
2..33	31:0	**Forward Quantizer Matrix**		
		Project:	All	
		Format:	U32	
		The format of a Quantizer Matrix is an 8x8 matrix in raster order. Each element is an unsigned byte.		

This is a frame-level state. Reciprocal Scaling Lists are always sent from the driver regardless whether they are specified by an application or the default/flat lists are being used. This is done to save the ROM (to store the default matrices) inside the PAK Subsystem. Hence, the driver is responsible for determining the final set of scaling lists to be used for encoding the current slice, based on the AVC Spec (Fall-Back Rules A and B). For encoding, there is no need to send the qm_list_flags[i], i=0 to7 and qm_present_flag to the PAK, since Scaling Lists syntax elements are encoded above Slice Data Layer.

FQM Reciprocal Scaling Lists elements are 16-bit each, conceptually equal to 1/ScaleValue. QM matrix elements are 8-bit each, equal to ScaleValue. However, in AVC spec., the Reciprocal Scaling Lists elements are not exactly equal to one-over of the corresponding Scaling Lists elements. The numbers are adjusted to simplify hardware implementation.

For all the description below, a scaling list set contains 6 4x4 scaling lists (or forward scaling lists) and 2 8x8 scaling lists (or forward scaling lists).

In MFX PAK mode, PAK needs both forward Q scaling lists and IQ scaling lists. The IQ scaling lists are sent as in MFD in raster scan order as shown in MFX_AVC_QM_STATE. But the Forward Q scaling lists are sent in transport form, i.e. column-wise raster order (column-by-column) to simplify the H/W. Driver will perform all the scan order conversion for both ForwardQ and IQ.

Precisely, if the reciprocal forward scaling matrix is F[4][4], then the 16 word of the matrix will be set as the following:

	bits 0-15	bits 16-31
DW0	F[0][0]	F[1][0]
DW1	F[2][0]	F[3][0]
DW2	F[0][1]	F[1][1]
DW3	F[2][1]	F[3][1]
DW4	F[0][2]	F[1][2]
DW5	F[2][2]	F[3][2]
DW6	F[0][3]	F[1][3]
DW7	F[2][3]	F[3][3]

2. AVC (H.264)

2.1 AVC Common Commands

The following commands are common for AVC decode and AVC encode.

2.1.1 MFX_AVC_IMG_STATE Command

MFX_AVC_IMG_STATE		
Source:		VideoCS
Length Bias:		2
This must be the very first command to issue after the surface state, the pipe select and base address setting commands. This command supports both Long and Short VLD and IT DXVA2 AVC Decoding Interface.		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_AVC_IMG_STATE Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h AVC_COMMON Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 0h Format: OpCode
	15:12	**Reserved** Project: All Format: MBZ
	11:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Project: All Format: =n 00Eh, used for normal decode and encode mode000h, a special case to provide a dummy image state for stitch mode operation. In this case, fields in DW1 which is part of the dummy image state command are ignored by hardware.
1	31:16	**Reserved** Project: All Format: MBZ

MFX_AVC_IMG_STATE

	15:0	**Frame Size** Project: All Format: U16-1 in MB unit The value for FrameSizeInMBs must match the product of FrameWidthInMBs and FrameHeightInMBs.Max. Screen resolution is therefore limited to 256 x 256 in MB unit. This parameter is specified for Intel interface only, not present in the DXVA. **Value** / **Name** / **Description** [0,16383] / / representing Number of MBs [1,16384]
2	31:24	**Reserved** Project: All Format: MBZ (bit[31:24] must be zero to match the DXVA 16-bit definition for FrameHeightInMBsMinus1)
	23:16	**Frame Height** Project: All Format: U8-1 in MB unit It is set to the value of (FrameHeightInMBsMinus1+ 1). Since the max value for FrameHeightInMBs is 255, the max allowed value for FrameHeightInMBsMinus1 is only 254. The min value for FrameHeightInMBs is 1.Although the max. value that can be specified for FrameHeightInMBs is 255 (in the current implementation), FrameWidthInMBs * FrameHeightInMBs must not exceed the max value of FrameSizeInMBs[14:0].e.g. for 1920x1080, FrameHeightInMBs[7:0] is equal to 68 (1080 divided by 16, and rounded up, i.e. effectively specified as 1088 instead).It is derived from FrameHeightInMbs = (2 – frame_mbs_only_flag) * PicHeightInMapUnits and PicHeightInMbs = FrameHeightInMbs / (1 + field_pic_flag) internally done. For MBAFF, PicHeightInMapUnits is in MB pair unit, so the bitstream sends only half frame height. **Value** / **Name** / **Description** [0,255] / / representing height [1,256]
	15:8	**Reserved** Project: All Format: MBZ (bit[15:8] must be zero to match the DXVA 16-bit definition for FrameWidthInMBsMinus1)
	7:0	**Frame Width** Project: All Format: U8-1 in MB unit It is set to the value of (FrameWidthInMBsMinus1+ 1). Since the max value for FrameWidthInMBs is 255, the max allowed value for FrameWidthInMBsMinus1 is only 254. The min value for FrameWidthInMBs is 1.Although the max. value that can be specified for FrameWidthInMBs is 255 (in the current implementation), FrameWidthInMBs * FrameWidthInMBs must not exceed the max value of FrameSizeInMBs[14:0].e.g. for 1920x1080, FrameHeightInMBs[7:0] is equal to 68 (1080 divided by 16, and rounded up, i.e. effectively specified as 1088 instead).It is derived from FrameWidthInMbs = (2 – frame_mbs_only_flag) * PicWidthInMapUnits and PicWidthInMbs = FrameWidthInMbs / (1 + field_pic_flag) internally done. For MBAFF, PicWidthInMapUnits is in MB pair unit, so the

bitstream sends only half frame width.

Value	Name	Description
[0,255]		representing width [1,256]

3	31:29	**Reserved** Project: All Format: MBZ (bit[31:29] must be zero to match the DXVA2 8-bit definition for InitQpChroma[1])
	28:24	**Second Chroma QP Offset** Project: All Signed integer value. It should be in the range of -12 to +12 (according to AVC spec).It specifies the offset for determining QP Cr from QP Y. It is set to the upper 5 bits of the value of the syntax element (Chroma_qp_offset[9:0]) read from the current active PPS.Chroma_qp_offset [4:0] – chroma_qp_offset_bits (from the current active PPS)Chroma_qp_offset [9:5] – second_chroma_qp_offset_bits
	23:21	**Reserved** Project: All Format: MBZ (bit[23:21] must be zero to match the DXVA2 8-bit definition for InitQpChroma[1])
	20:16	**First Chroma QP Offset** Project: All Signed integer value. It should be in the range of -12 to +12 (according to AVC spec).It specifies the offset for determining QP Cb from QP Y. It is set to the lower 5 bits of the value of the syntax element (Chroma_qp_offset[9:0]) read from the current active PPS.Chroma_qp_offset [4:0] – chroma_qp_offset_bits (from the current active PPS)Chroma_qp_offset [9:5] – second_chroma_qp_offset_bits
	15:13	**Reserved** Project: All Format: MBZ
	12	**Weighted_Pred_Flag** Project: All (This field is defined differently from Gen6, Gen7 follows strictly DXVA2 AVC interface.)

Value	Name
0	specifies that weighted prediction is not used for P and SP slices **[Default]**
1	specifies that weighted prediction is used for P and SP slices

Programming Notes
This field must set to '0' for B and I pictures.

	11:10	**Weighted_BiPred_Idc** Project: All (This field follows strictly DXVA2 AVC interface.)

Value	Name
0	Specifies that the default weighted prediction is used for B slices **[Default]**
1	Specifies that explicit weighted prediction is used for B slices
2	Specifies that implicit weighted prediction is used for B slices.

		3 Illegal value
		Programming Notes This field must set to 0 for P and I pictures.
	9:8	**ImgStruct – Image Structure, img_structure[1:0]** Project: All The current encoding picture structure can only takes on 3 possible values

Value	Name
00b	Frame Picture
01b	Top Field Picture
11b	Bottom Field Picture
10b	Invalid, not allowed.

Programming Notes
img_structure[0] can be used as a flag to distinguish between frame and field structure. It must be consistent with the field_pic_flag setting in the Slice Header.This parameter is specified for Intel interface only, not present in the DXVA as a separate state (instead the img_structure[1] is embedded inside the DXVA picture definition).

	7:0	**Reserved** Project: All Format: MBZ
4	31:16	**MinFrameWSize** Default Value: 0h Project: All Format: U16

Minimum Frame Size [15:0] (in Word, 16-bit)(Encoder Only) Mininum Frame Size is specified to compensate for intel Rate Control Currently zero fill (no need to perform emulation byte insertion) is done only to the end of the CABAC_ZERO_WORD insertion (if any) at the last slice of a picture. Intel encoder parameter, not part of DXVA. The caller should always make sure that the value, represented by Mininum Frame Size, is always less than maximum frame size **FrameBitRateMax (DWORD 10 bits** 29:16).This field is reserved in Decode mode.

The programmable range 0…2^18-1

When MinFrameWSizeUnits is 00.

Programmable range is 0…2^20-1 when MinFrameWSizeUnits is 01.

Programmable range is 0…2^26-1 when MinFrameWSizeUnits is 10.

Programmable range is 0…2^32-1 when MinFrameWSizeUnits is 11.

	15	**MbStatEnabled** Project: All Enable reading in MB status buffer (a.k.a. encoding stream-out buffer) Note: For multi-pass encoder, all passes except the first one need to set this value to 1. By setting the first pass to 0, it does save some memory bandwidth.

Value	Name	Description	Project

0	Disable	Disable Reading of Macroblock Status Buffer	All
1	Enable	Enable Reading of Macroblock Status Buffer	All

14 **LoadSlicePointerFlag**

Project: All

LoadBitStreamPointerPerSlice (Encoder-only)To support multiple slice picture and additional header/data insertion before and after an encoded slice.When this field is set to 0, bitstream pointer is only loaded once for the first slice of a frame. For subsequent slices in the frame, bitstream data are stitched together to form a single output data stream.When this field is set to 1, bitstream pointer is loaded for each slice of a frame. Basically bitstream data for different slices of a frame will be written to different memory locations.

Value	Name	Description	Project
0	Disable	Load BitStream Pointer only once for the first slice of a frame	All
1	Enable	Load/reload BitStream Pointer only once for the each slice, reload the start location of the bitstream buffer from the Indirect PAK-BSE Object Data Start Address field	All

12 **MvUnpackedFlag**

Project: All

MVUnPackedEnable (Encoder Only)This field is reserved in Decode mode.

Value	Name	Description	Project
0	use packed MV format (compliant to DXVA)	Desc	All
1	use unpacked 8MV/32MV format only	Desc	All

11:10 **ChromaFormatIdc**

Project: All

Chroma Format IDC, ChromaFormatIdc[1:0]It specifies the sampling of chroma component (Cb, Cr) in the current picture as follows :

Value	Name	Description	Project
00b	monochrome picture	Desc	All
01b	4:2:0 picture	Desc	All
10b	4:2:2 picture (not supported)		
11b	4:4:4 picture (not supported)		

Programming Notes

It is set to the value of the syntax element read from the current active SPS.The corresponding Monochrome Flag (monochrome_flag) can be derived from this field.

9 **Reserved**

Project: All

Format: MBZ

8 **MbMvFormatFlag**

Project: All

Use MB level MvFormat flag (Encoder Only)(This bit must be set to zero in IVB:GT2:A0)

Value	Name	Description	Project
0	HW PAK ignore MvFormat in the MB data.	When bit 12 == 0, all MBs use packed MV formatWhen bit 12 == 1, each MB data must use unpacked MV format, 8MV when there is no minor MV involved, and 32MV if there are some minor MVs.	All
1	(not for [SNB], [IVB] only) HW PAK will follow		All

MFX_AVC_IMG_STATE

	MvFormat value set within each MB data.	

Programming Notes

They must take one of the two values: the 8MV unpacked format (MvFormat =101b), and the 32MV unpacked format (MvFormat =110b).This bit can be set only when MvUnpackedFlag (bit 12 of this register) is set otherwise system could hang.

7 **EntropyCodingFlag**

Project: All

Entropy Coding Flag, entropy_coding_flag

Value	Name	Description	Project
0	CAVLC bit-serial encoding mode	Desc	All
1	CABAC bit-serial encoding mode.	Desc	All

Programming Notes

It specifies one of the two possible bit stream encoding modes in the AVC. It is set to the value of the syntax element read from the current active PPS.

6 **ImgDisposableFlag**

Project: All

Current Img Disposable Flag or Non-Reference Picture Flag

Value	Name	Description	Project
0	the current decoding picture may be used as a reference picture for others	Desc	All
1	the current decoding picture is not used as a reference picture (e.g. a B-picture cannot be a reference picture for any subsequent decoding)	Desc	All

Programming Notes

It is derived from ImgDisposableFlag = (nal_ref_idc == 0). nal_ref_idc is a syntax element from a NAL unit. When this flag is set, no reference picture and DMV are written out.This field is only valid for VLD decoding mode.

5 **ConstrainedIPredFlag**

Project: All

Constrained Intra Prediction Flag, constrained_ipred_flagIt is set to the value of the syntax element in the current active PPS.

Value	Name	Description	Project
0	allows both intra and inter neighboring MB to be used in the intra-prediction encoding of the current MB.	Desc	All
1	allows only to use neighboring Intra MBs in the intra-prediction encoding of the current MB. If the neighbor is an inter MB, it is considered as not available.	Desc	All

4 **Direct8x8InfFlag**

Project: All

Direct 8x8 Inference Flag, direct_8x8_inference_flagIt is set to the value of the syntax element in the current active SPS.It specifies the derivation process for luma motion vectors in the Direct MV coding modes (B_Skip, B_Direct_16x16 and B_Direct_8x8). When frame_mbs_only_flag is equal to 0, direct_8x8_inference_flag shall be equal to 1.It must be

consistent with the frame_mbs_only_flag and transform_8x8_mode_flag settings.

Value	Name	Description	Project
0	allows subpartitioning to go below 8x8 block size (i.e. 4x4, 8x4 or 4x8)	Desc	All
1	allows processing only at 8x8 block size. MB Info is stored for 8x8 block size.	Desc	All

3 **Transform8x8Flag**

Project:	All

8x8 IDCT Transform Mode Flag, trans8x8_mode_flagSpecifies 8x8 IDCT transform may be used in this pictureIt is set to the value of the syntax element in the current active PPS.

Value	Name	Description	Project
0	no 8x8 IDCT Transform, only 4x4 IDCT transform blocks are present	Desc	All
1	8x8 Transform is allowed	Desc	All

2 **FrameMbOnlyFlag**

Project:	All

Frame MB only flag, frame_mbs_only_flagIt is set to the value of the syntax element in the current active SPS.

Value	Name	Description	Project
0	not true ; effectively enables the possibility of MBAFF mode.	Desc	All
1	true, only frame MBs can occur in this sequence, hence disallows the MBAFF mode and field picture.	Desc	All

1 **MbaffFlameFlag**

Project:	All

MBAFF mode is active, mbaff_frame_flag.It is derived from MbaffFrameFlag = (mb_adaptive_frame_field_flag && ! field_pic_flag). mb_adaptive_frame_field_flag is a syntax element in the current active SPS and field_pic_flag is a syntax element in the current Slice Header. They both are present only if frame_mbs_only_flag is 0. Although mbaff_frame_flag is a Slice Header parameter, its value is expected to be the same for all the slices of a picture.It must be consistent with the mb_adaptive_frame_field_flag, the field_pic_flag and the frame_mbs_only_flag settings.This bit is valid only when the img_structure[1:0] indicates the current picture is a frame.

Value	Name	Description	Project
0	not in MBAFF mode	Desc	All
1	in MBAFF mode	Desc	All

0 **FieldPicFlag**

Project:	All

Field picture flag, field_pic_flag, specifies the current slice is a coded field or not.It is set to the same value as the syntax element in the Slice Header. It must be consistent with the img_structure[1:0] and the frame_mbs_only_flag settings.Although field_pic_flag is a Slice Header parameter, its value is expected to be the same for all the slices of a picture.

Value	Name	Project
0h	a slice of a coded frame	All
1h	a slice of a coded field	All

5 31:17 Reserved

[ExistsIf]Encode Only 16 **NonFirstPassFlag**

This signals the current pass is not the first pass. It will imply designate HW behavior: e.g

Value	Name	Description	Project

	0h	Disable	Always use the MbQpY from initial PAK inline object for all passes of PAK	All
	1h	Enable	Use MbQpY from stream-out buffer if MbRateCtrlFlag is set to 1	All

15:13 **Reserved**

Project:	All
Format:	MBZ

12 **InterMbZeroCbpFlag – InterMB Force CBP to Zero Control Flag**

Inter MB Force CBP ZERO mask.

Value	Name	Description
0h	Disable	No Effect
1h	Enable	Zero out all A/C coefficients for the inter MB violating Inter Confirmance

11:10 **MinFrameWSizeUnits**

This field is the Minimum Frame Size Units

Value	Name	Description	Project
00b	compatibility mode	Minimum Frame Size is in old mode (words, 2bytes)	All
01b	16 byte	Minimum Frame Size is in 16bytes	All
10b	4Kb	Minimum Frame Size is in 4Kbytes	All
11b	16Kb	Minimum Frame Size is in 16Kbytes	All

9 **MbRateCtrlFlag – MB level Rate Control Enabling Flag**

MB Rate Control conformance mask

Value	Name	Description	Project
0h	Disable	Apply accumulative delta QP for consecutive passes on top of the macroblock QP values in inline data	All
1h	Enable	Apply RC QP delta to suggested QP values in Macroblock Status Buffer except the first pass.	All

Programming Notes
This field is ignored when MacroblockStatEnable is disabled or MB level Rate control flag for the current MB is disable in Macroblock Status Buffer.

8 **Reserved**

Project:	All
Format:	MBZ

7 **Intra/InterMbIpcmFlag – ForceIPCMControlMask**

This field is to Force **IPCM** for Intra or Inter Macroblock size conformance mask.

Value	Name	Description	Project
0h	Disable	Do not change intra macroblocks even.	
1h	Enable	Change intra macroblocks MB_type to IPCM.	

Programming Notes
This field is ignored when MacroblockStatEnable is disabled or MB level Intra MB conformance flag for the current MB is disable in Macroblock Status Buffer.

6:4 **Reserved**

Project:	All
Format:	MBZ

3 **FrameSzUnderFlag – FrameBitRateMinReportMask**

MFX_AVC_IMG_STATE

This is a mask bit controlling if the condition of frame level bit count is less than FrameBitRateMin

Value	Name	Description	Project
0h	Disable	Do not update bit0 of MFC_IMAGE_STATUS control register.	All
1h	Enable	set bit0 and bit 1of MFC_IMAGE_STATUS control register if the total frame level bit counter is less than or equal to Frame Bit rate Minimum limit.	All

2 **FrameSzOverFlag – FrameBitRateMaxReportMask**

This is a mask bit controlling if the condition of frame level bit count exceeds FrameBitRateMax.

Value	Name	Description	Project
0	Disable	Do not update bit0 of MFC_IMAGE_STATUS control register.	All
1	Enable	Set bit0 and bit 1 of MFC_IMAGE_STATUS control register if the total frame level bit counter is greater than or equal to Frame Bit rate Maximum limit.	All

1 **InterMbMaxBitFlag – InterMBMaxSizeReportMask**

This is a mask bit controlling if the condition of any inter MB in the frame exceeds InterMBMaxSize.

Value	Name	Description	Project
0	Disable	Do not update bit0 of MFC_IMAGE_STATUS control register.	All
1	Enable	Set bit0 of MFC_IMAGE_STATUS control register if the total bit counter for the current MB is greater than the Inter MB Conformance Max size limit.	All

0 **IntraMbMaxBitFlag – IntraMBMaxSizeReportMask**

This is a mask bit controlling if the condition of any intra MB in the frame exceeds IntraMBMaxSize.

Value	Name	Description	Project
0h	Disable	Do not update bit0 of MFC_IMAGE_STATUS control register.	All
1	Enable	set bit0 of MFC_IMAGE_STATUS control register if the total bit counter for the current MB is greater than the Intra MB Conformance Max size limit.	All

6 [ExistsIf]Encode Only

31:28 **Reserved**

27:16 **InterMbMaxSz**

Project:	All
Format:	U12

This field, Inter MB Conformance Max size limit,indicates the allowed max bit count size for Inter MB

15:12 **Reserved**

Project:	All
Format:	MBZ

11:0 **IntraMbMaxSz**

Project:	All
Exists If:	Intra Only
Format:	U12

This field, Intra MB Conformance Max size limit,indicates the allowed max bit count size for Intra MB

MFX_AVC_IMG_STATE

		All IPCM MBs should ignore this Max size limit.
7	31:0	**Reserved**
8 [ExistsIf]Encode Only	31:24	**SliceDeltaQpMax[3]** Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP for total bit-count above FrameBitRateMax - first 1/8 regionThis field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame exceeds FrameBitRateMax but is within 1/8 of FrameBitRateMaxDelta above FrameBitRateMax, i.e., in the range of (FrameBitRateMax, (FrameBitRateMax+ FrameBitRateMaxDelta>>3).
	23:16	**SliceDeltaQpMax[2]** Project: All Format: U8 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP for bit-count above FrameBitRateMax - above 1/8 and below 1/ 4 This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between 1/8 and ¼ of FrameBitRateMaxDelta above FrameBitRateMax, i.e., in the range of ((FrameBitRateMax+ FrameBitRateMaxDelta>>3), (FrameBitRateMax+ FrameBitRateMaxDelta>>2).
	15:8	**SliceDeltaQpMax[1]** Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP for bit-count above FrameBitRateMax – above1/ 4 and below 1/2 This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between ¼ and ½ of FrameBitRateMaxDelta above FrameBitRateMax, i.e., in the range of ((FrameBitRateMax+ FrameBitRateMaxDelta>>2), (FrameBitRateMax+ FrameBitRateMaxDelta>>1).
	7:0	**SliceDeltaQpPMax[0]** Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP for bit-count above FrameBitRateMax - above 1/ 2This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is above FrameBitRateMax by more than half the distance of FrameBitRateMaxDelta , i.e., in the range of ((FrameBitRateMax+ FrameBitRateMaxDelta>>1), infinite).
9 [ExistsIf]Encode Only	31:24	**SliceDeltaQpMin[3]** Format: S7 Range: [0:MAX_QP_DELTA]

		This field is the Slice level delta QP for total bit-count below FrameBitRateMin - first 1/8 regionThis field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is less than FrameBitRateMin and greater than or equal to 1/8 the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of [(FrameBitRateMin- FrameBitRateMinDelta>>3), FrameBitRateMin).
	23:16	**SliceDeltaQpMin[2]** Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP for bit-count below FrameBitRateMin – below 1/ 8 and above 1/ 4This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between one-eighth and quarter the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of [(FrameBitRateMin- FrameBitRateMinDelta>>2), (FrameBitRateMin- FrameBitRateMinDelta>>3)).
	15:8	**SliceDeltaQpMin[1]** Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP for bit-count below FrameBitRateMin– below 1/4 and above 1/ 2This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between quarter and half the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of [(FrameBitRateMin- FrameBitRateMinDelta>>1), (FrameBitRateMin- FrameBitRateMinDelta>>2)).
	7:0	**SliceDeltaQpMin[0]** Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice Level Delta QP for bit-count below FrameBitRateMin – below 1/ 2This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is below FrameBitRateMin by more than half the distance of FrameBitRateMinDelta , i.e., in the range of [0, (FrameBitRateMin- FrameBitRateMinDelta>>1).
10 [ExistsIf]Encode Only	31	**FrameBitrateMaxUnit** This field is the Frame Bitrate Maximum Limit Units.

Value	Name	Description	Project
0	Byte	FrameBitRateMax is in units of 32 Bytes when FrameBitrateMaxUnitMode is 1 and in units of 128 Bytes if FrameBitrateMaxUnitMode is 0	All
1	Kilo Byte	FrameBitRateMax is in units of 4KBytes Bytes when FrameBitrateMaxUnitMode is 1 and in units of 16KBytes if FrameBitrateMaxUnitMode is 0	All

	30	**FrameBitrateMaxUnitMode** This field is the Frame Bitrate Maximum Limit Units.

Value	Name	Description	Project
0h	compatibility mode	FrameBitRateMaxUnit is in old mode (128b/16Kb)	All

MFX_AVC_IMG_STATE

		1h	New mode	FrameBitRateMaxUnit is in new mode (32byte/4Kb)	All
	29:16	**FrameBitRateMax** This field is the Frame Bitrate Maximum Limit. This field along with FrameBitrateMaxUnit determines maximum allowed bits in a frame before multi-pass gets triggered (when enabled). In other words, multi-pass is triggered when the actual frame byte count exceeds this value. When FrameBitrateMaxUnitMode is 0(compatibility mode) bits 16:27 should be used, bits 28 and 29 should be 0.. **Value** / **Name** / **Description** 0-512KB / / The programmable range is 0-512KB when FrameBitrateMaxUnit is 0. 0-8190KB / / The programmable range is 0-8190KB when FrameBitrateMaxUnit is 1.			
	15	**FrameBitrateMinUnit** This field is the Frame Bitrate Minimum Limit Units. **Value** / **Name** / **Description** / **Project** 0 / Byte / FrameBitRateMax is in units of 32 Bytes when FrameBitrateMinUnitMode is 1 and in units of 128 Bytes if FrameBitrateMinUnitMode is 0 / All 1 / Kilo Byte / FrameBitRateMax is in units of 4KBytes Bytes when FrameBitrateMaxUnitMode is 1 and in units of 16KBytes if FrameBitrateMaxUnitMode is 0 / All			
	14	**FrameBitrateMinUnitMode** This field is the Frame Bitrate Minimum Limit Units. **Value** / **Name** / **Description** / **Project** 0h / Compatibility mode / FrameBitRateMaxUnit is in old mode (128b/16Kb) / All 1h / New mode / FrameBitRateMaxUnit is in new mode (32byte/4Kb) / All			
	13:0	**FrameBitRateMin** RangeThe programmable range 0-512KB When FrameBitrateMinUnit is in 0.Programmable range is 0–8190 KB when FrameBitrateMinUnit is in 1.This field is the Frame Bitrate Minimum Limit ()This field along with FrameBitrateMinUnit determines minimum allowed bits in a Frame before Multi-Pass gets triggered (when enabled). In other words, multi-pass is triggered when the actual frame byte count is less than this value. When FrameBitrateMinUnitMode is 0 (compatibility mode) bits 0:11 should be used, bits 12 and 13 should be 0.			
11 [ExistsIf]Encode Only	31	**Reserved**			
	30:16	**FrameBitRateMaxDelta** Project: All Exists If: Always Format: U15 This field is used to select the slice delta QP when FrameBitRateMax Is exceeded. It shares the same FrameBitrateMaxUnit. When FrameBitrateMaxUnitMode is 0(compatibility mode) bits 16:27 should be used, bits 28, 29 and 30 should be 0. **Value** / **Name** / **Description** 0-1024KB / / The Programmable range 0-1024KB when FrameBitRateMaxUnit is 0. 0-16380KB / / The Programmable range is 0–16380KB when FrameBitRateMaxUnit is 1. 0h / **[Default]** /			
	15	**Reserved** Project: All Format: MBZ			
	14:0	**FrameBitRateMinDelta** Range: The programmable range 0-1024KB When FrameBitrateMinUnit is in			

		32Bytes.Programmable range is 0–16380KB when FrameBitrateMinUnit is in 4Kbytes.
		This field is used to select the slice delta QP when FrameBitRateMin Is exceeded. It shares the same FrameBitrateMinUnit. When FrameBitrateMinUnitMode is 0(compatibility mode) bits 0:11 should be used, bits 12, 13 and 14 should be 0.Note: HW requires the following condition FrameBitRateMinDelta <= 2*FrameBitRateMinMust be true, otherwise it may cause unpredicted behavior.
12	31:0	**Reserved**
		Project: All
		Format: MBZ
13	31:30	**Reserved**
		Project: All
		Format: MBZ
	29	**Current Picture Has Performed MMCO5**
		Set to 1 if the current Pic has performed the memory_management_control_operation = = 5.
	28:24	**Number of Reference Frames**
		Format: U5
		Range: Range 0 to MaxDpbSize (=16 for Level 4.1)
		Specifies the maximum number of reference frames (frames, field pairs, unpaired field) existed in the current DBP for decoding the current picture.
	23:22	**Reserved**
		Project: All
		Format: MBZ
	21:16	**Number of Active Reference Pictures from L1**
		Format: U6-1
		Specifies the initial maximum reference index value minus 1 to access the L1 Reference List. It is extracted from PPS. It corresponds to the number of active reference pictures from L1 to decode the current picture. It can be modified by the slice header if num_ref_idx_active_override_flag is set. Only valid for B picture.
		Value / **Name**
		[0,31]
	15:14	**Reserved**
		Project: All
		Format: MBZ
	13:8	**Number of Active Reference Pictures from L0**
		Format: U6-1
		Specifies the initial maximum reference index value minus 1 to access the L0 Reference List. It is extracted from PPS. It corresponds to the number of active reference pictures from L0 to decode the current picture. It can be modified by the slice header if num_ref_idx_active_override_flag is set. Valid for both P and B pictures.
		Value / **Name**
		[0,31]
	7:0	**Initial QP Value**
		Format: S7
		Description / **Project**

MFX_AVC_IMG_STATE			
		Range: [-26,25]	
		Short Format Only	
		Initial QP value for a Slice, extracted from PPS. It may further get modified by slice_qp_delta in slice header and mb_qp_delta in MB header.	

MAX_QP_DELTA : Maximum QP delta is the Magnitude of QP delta between passes.

MAX_QP_DELTA is selected such that cumulative QP over all possible passes shouldn't exceed 51.

Example Configurations:

MAX Number of Passes	MAX_QP_DELTA
4	0xc
5	0xa
6	0x8
7	0x7

2.1.2 MFX_AVC_DIRECTMODE_STATE Command

MFX_AVC_DIRECTMODE_STATE			
Source:		VideoCS	
Length Bias:		2	
This is a slice level command and can be issued multiple times within a picture that is comprised of multiple slices. All DMV buffers are treated as standard media surfaces, in which the lower 6 bits are used for conveying surface states.Current Pic POC number is assumed to be available in POCList[32 and 33] of the MFX_AVC_DIRECTMODE_STATE Command.This command is only valid in the AVC decoding in VLD and IT modes, and AVC encoder mode. The same command supports both Long and Short DXVA2 AVC Interface.			
DWord	**Bit**	**Description**	
0	31:29	**Command Type**	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	**Pipeline**	
		Default Value:	2h MFX_SINGLE_DW
		Format:	OpCode
	26:24	**Media Command Opcode**	
		Default Value:	1h AVC
		Format:	OpCode
	23:21	**SubOpcodeA**	
		Default Value:	0h MEDIA_
		Format:	OpCode
	20:16	**SubOpcodeB**	
		Default Value:	2h Desc
		Format:	OpCode
	15:12	**Reserved**	
		Project:	All

MFX_AVC_DIRECTMODE_STATE

		Format: MBZ
	11:0	**DWord Length** Default Value: 0043h Excludes DWord (0,1) Project: All Format: =n Total Length - 2
1	31:6	**Direct MV Buffer Base Address for Picture 0 (current or reference top field)** This field provides the base address of the DMV write buffer to store motion vectors decoded in the current picture (top field), which may be used later as a collocated motion information read buffer of the associated reference picture in decoding subsequent B-pictures that have MB coded in direct mode. It is a private buffer used by the MPR hardware only. Its content is not accessed by software.This buffer must be 64-byte cacheline aligned.The write buffer size is 557,056 bytes for 1 frame. Scalable with frame height, but do not scale with frame width as the hardware assumes frame width (in MBs) fixed at 128 (smallest power of 2 value larger than 120 – 1920x1088 screen resolution)It is only valid if the current picture is a progressive frame, MbAff frame, or a top field.There are a total of 32 reference picture (previously decoded) Direct MV Buffers (0 to 31, not including the DMV write buffer 32 and 33 of the current picture) to read in the corresponding collocated DMV and motion information. For reference picture, these 32 DMV read Buffers can be indexed by the frame_store_ID[4:0], which is obtained from RefPicList L0/L1[RefPicIdx]. frame_Store_IDbit[0] (indicator for Top/Bottiom Field).For writing out motion information during the decoding of the current picture, all 34 DMV buffers can be addressed by [img_dec_fs_idc[4:0]<<1 + img_structure[1]].
	5:4	**Direct MV Buffer – Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.

Value	Name	Description	Project
00b	Highest priority	Desc	All
01b	Second highest priority	Desc	All
10b	Third highest priority		
11b	Lowest priority		

Programming Notes
This field of Picture 0 DMV Buffer must always be programmed, regardless if this buffer is active or not, exist or not. H/W only reads this bit to determine the arbitration priority control for all 34 possible DMV buffers. This field is ignored in all the other DMV buffers 1 to 33.

	2	**Direct MV Buffer - Graphics Data Type (GFDT) for Picture 0** This field contains the GFDT bit for this surface when writes occur. GFDT can also be set by the GTT. The effective GFDT is the logical OR of this field with the GFDT from the GTT entry. This field is ignored for reads.

Value	Name	Description	Project
0h	Disable	Desc	All
1h	Enable	Desc	All

Programming Notes
This field of Picture 0 DMV Buffer must always be programmed, regardless if this buffer is active or not, exist or not. H/W only reads this bit to determine the GFDT for all 34 possible DMV buffers. This field is ignored in all the other DMV buffers 1 to 33.

	1:0	**Direct MV Buffer - Cacheability Control for Picture 0** Format: U2 Enumerated Type

MFX_AVC_DIRECTMODE_STATE

This field controls cacheability in the mid-level cache (MLC) and last-level cache (LLC).

Value	Name	Description	Project
00b	Use cacheability control bits from GTT entry	Desc	All
01b	Data is not cached in LLC or MLC	Desc	All
10b	Data is cached in LLC but not MLC		
11b	Data is cached in both LLC and MLC		

Programming Notes

This field of Picture 0 DMV Buffer must always be programmed, regardless if this buffer is active or not, exist or not. H/W only reads this bit to determine the cacheability control for all 34 possible DMV buffers. This field is ignored in all the other DMV buffers 1 to 33.

2 | 31:6 | **Direct MV Buffer Base Address for Picture 1 (current or reference bottom field)**

This field provides the base address of the DMV read/write buffer for the current or reference picture (bottom field). It is paired with the DMV Buffer of Picture 0 for MB pair retrieval during read.It follows the same format specification as DMV buffer for Picture 0It is only valid if the current picture is a bottom field. It is also valid

5:4 | **Direct MV Buffer – Arbitration Priority Con**

Project:	All
Format:	U2

This field is ignored in H/W, and assumes the same value as of Picture 0 DMV Buffer specification bit[5:4] above.

2 | **Direct MV Buffer - Graphics Data Type (GFDT) for Picture 1**

Project:	All
Format:	U1

This field is ignored in H/W, and assumes the same value as of Picture 0 DMV Buffer specification bit[2] above.

1:0 | **Direct MV Buffer -Cacheability Control for Picture 1**

Project:	All
Format:	U2

This field is ignored in H/W, and assumes the same value as of Picture 0 DMV Buffer specification bit[1:0] above.

3..32 | 31:6 | **Direct MV Buffer Base Address for Reference Frame 2 to 31**

This field provides the base address of the DMV buffer for reference frame 2 to 31. They are needed if the current B-Picture has MBs coded in direct mode. It is a private buffer used by the MPR hardware only. Its content is not accessed by software.All these buffers must be 64-byte cacheline aligned.There are a total of 32 possible Direct MV Read Buffers (not including the current write buffer of the current picture) to read in the corresponding DMV. Each read buffer size is 557,056 bytes for 1 frame (the selected colPic). Scalable with frame height, but do not scale with frame width as the hardware assumes frame width (in MBs) fixed at 128 (smallest power of 2 value larger than 120 – 1920x1088 screen resolution).The adjacent DMV buffers are paired ([2 and 3], [4 and 5], [N and N+1], ..[30 and 31]).

5:4 | **Direct MV Buffer – Arbitration Priority Control**

Project:	All
Format:	U2

This field is ignored in H/W, and assumes the same value as of Picture 0 DMV Buffer specification

		bit[5:4] above.
	2	**Direct MV Buffer - Graphics Data Type (GFDT) for Reference Frame 2 to 31** Project: All Format: U1 This field is ignored in H/W, and assumes the same value as of Picture 0 DMV Buffer specification bit[2] above.
	1:0	**Direct MV Buffer - Cacheability Control for Reference Frame 2 to 31** Project: All Format: U2 This field is ignored in H/W, and assumes the same value as of Picture 0 DMV Buffer specification bit[1:0] above.
33..34	31:6	**Direct MV Buffer Base Addresses 32 and 33 (Write-Only Buffer), for Current Decoding Frame/Field** This field provides the base address of the DMV write-only buffer for the current decoding frame/field.It is a private buffer used by the MPR hardware only. Its content is not accessed by software.All these buffers must be 64-byte cacheline aligned, i.e. the same as the above DMV read/write buffers.These 2 buffers can only be addressed by [img_dec_fs_idc[4:0]<<1 + img_structure[1]] for the current picture being decoded.Each write buffer size is 557,056 bytes for 1 frame (the selected colPic). Scalable with frame height, but do not scale with frame width as the hardware assumes frame width (in MBs) fixed at 128 (smallest power of 2 value larger than 120 – 1920x1088 screen resolution).DMV write buffer 32 is valid only if the current picture is a progressive frame, MbAff frame, or a top field.DMV write buffer 33 is valid only if the current picture is a bottom field.
	5:4	**Direct MV Buffer 32 and 33 (Write-only Buffer) – Arbitration Priority Control** Project: All Format: U2 This field is ignored in H/W, and assumes the same value as of Picture 0 DMV Buffer specification bit[5:4] above.
	3	**Reserved** This field is ignored for writes.
	2	**Direct MV Buffer 32 and 33 (Write-only Buffer) - Graphics Data Type (GFDT) for Current Frame/Field** Project: All Format: U1 This field is ignored in H/W, and assumes the same value as of Picture 0 DMV Buffer specification bit[2] above.
	1:0	**Direct MV Buffer 32 and 33 (Write-only Buffer) - Cacheability Control for Current Frame/Field** Project: All Format: U2 This field is ignored in H/W, and assumes the same value as of Picture 0 DMV Buffer specification bit[1:0] above.
35..68	31:0	**POC List, POCList[34][31:0]** Each POC value is a signed 32-bit number.One-to-one correspondence with the 34 Direct MV Buffer Address for Reference and Currrent Frames/FieldsThere are 34 POC entries in the list. For reference picture, only the lower 32 POC [0-31] entries can be used, and POCList[ ] is indexed by the

MFX_AVC_DIRECTMODE_STATE		
		frame_store_ID[4:0], which is obtained from RefPicList L0/L1[RefPicIdx]. frame_Store_IDbit[0] (indicator for Top/Bottiom Field).For current picture, all 34 POC entries [0-33] can be addressed by POCList[img_dec_fs_idc[4:0]<<1 + img_structure[1]].For frame-only mode, every other entry is skipped. For MBAFF and field-only picture, each entry is a field POC, and every two entries are paired.

2.1.3 MFX_AVC_SLICE_STATE Command

MFX_AVC_SLICE_STATE			
Source:		VideoCS	
Length Bias:		2	
This is a slice level command and can be issued multiple times within a picture that is comprised of multiple slices. The same command is used for AVC encoder (PAK mode) and decoder (VLD and IT modes).			
Programming Notes			
MFX_AVC_SLICE_STATE command is not issued for AVC DXVA2 Short Format Bitstream decode, instead MFD_AVC_SLICEADDR command is executed to retrieve the next slice MB Start Address X and Y by H/W itself.			
DWord	**Bit**	**Description**	
0	31:29	**Command Type**	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	**Pipeline**	
		Default Value:	2h MFX_AVC_SLICE_STATE
		Format:	OpCode
	26:24	**Command Opcode**	
		Default Value:	1h AVC
		Format:	OpCode
	23:21	**SubOpcodeA**	
		Default Value:	0h MFX_AVC_SLICE_STATE
		Format:	OpCode
	20:16	**Command SubOpcodeB**	
		Default Value:	3h MFX_AVC_SLICE_STATE
		Format:	OpCode
	15:12	**Reserved**	
		Format:	MBZ
	11:0	**DWord Length**	
		Default Value:	8h DWORD_COUNT_n
		Excludes DWords 0,1	
1	31:4	**Reserved**	
		Format:	MBZ
	3:0	**Slice Type**	
		It is set to the value of the syntax element read from the Slice Header.	
		Value	**Name**
		0000b	P Slice
		0001b	B Slice

MFX_AVC_SLICE_STATE

		0010b: I Slice 0011b-1111b: Reserved **Programming Notes** Bits[3:2] must be 0
2	31:30	**Reserved** Format: MBZ
	29:24	**Number of Reference Pictures in Inter-prediction List 1** Format: U6 This field is valid only for encoding a B Slice, for which it is expected to have at least one entry in the reference list L1; otherwise (if Slice Type is not a B Slice), this field must be set to 0. This field can be derived for a B Slice from the Slice Header syntax element NumRefIdxActiveMinus1 as, Num_Ref_Idx_L1 = NumRefIdxActiveMinus1[1] + 1. **Value**: 0-32; **Name**:
	23:22	**Reserved** Format: MBZ
	21:16	**Number of Reference Pictures in Inter-prediction List 0** Format: U6 This field is valid for encoding a P or B Slice, for which it is expected to have at least one entry in the reference list L0; otherwise (if Slice Type is not a P or B Slice), this field must be set to 0. This field can be derived for a P or B Slice from the Slice Header syntax element NumRefIdxActiveMinus1 as, Num_Ref_Idx_L0 = NumRefIdxActiveMinus1[0] + 1. **Value**: 0-32; **Name**:
	15:11	**Reserved** Format: MBZ
	10:8	**Log 2 Weight Denom Chroma** Format: U3 **Value**: 0-7; **Name**:
	7:3	**Reserved** Format: MBZ
	2:0	**Log 2 Weight Denom Luma** Format: U3 It is the base 2 logarithm of the denominator for all Luma weighting factors. It is set to the value of the syntax element read from the Slice Header Pred_Weight_Table(). **Value**: 0-7; **Name**:
3	31:30	**Weighted Prediction Indicator** This field indicates the Weighted Prediction mode for a P or B Slice. It is a combined field corresponding to the syntax element WeightedBiPredIdc or WeightedPredFlag read from the current active PPS.

If it is a B-Slice, these bits are interpreted as:

00b – Specifies the default weighted inter-prediction to be applied
01b – Specifies the explicit weighted inter-prediction to be applied
10b – Specifies the implicit weighted inter-prediction to be applied
11b – Reserved (not allowed)

If it is a P Slice, these bits are interpreted as:

00b – Disables weighted inter-prediction (Default weighted)
01b – Enables weighted inter-prediction (Explicit weighted)
10b - 11b – Reserved

Programming Notes
Only when in B Slice with Weighted_Pred_Idc = 1 (explicit weighted prediction), will there be a L1 and/or a L0 weight+offset tables being sent to the BSD unit through the Slice_State command. Only when in P Slice with Weighted_Pred_Idc = 1, will there be a L0 weight+offset table being sent to the BSD.
If Weighted_Pred_Idc != 1 for B Slice or Weighted_Pred_Idc =0 for P Slice, no Slice_State command should be issued to send these tables. If still being issued, the data is read but ignored.
DXVA specifies Weighted_Bipred and Weighted_Pred in frame-level state. However, these two flags are combined and specified in slice level for both P and B slice type.

29 **Direct Prediction Type**

Type of direct prediction used for B Slices. This field is valid only for Slice_Type = B Slice; otherwise, it must be set to 0.

Value	Name
0	Temporal
1	Spatial

28:27 **Disable Deblocking Filter Indicator**

Value	Name	Description
00b		FilterInternalEdgesFlag is set equal to 1
01b		Disable all deblocking operation, no deblocking parameter syntax element is read; filterInternalEdgesFlag is set equal to 0
10b		Macroblocks in different slices are considered not available; filterInternalEdgesFlag is set equal to 1
11b	Reserved	Not defined in AVC

26 **Reserved**

Format:	MBZ

25:24 **Cabac Init Idc[1:0]**

Specifies the index for determining the initialization table used in the context variable initialization process.

Value	Name
0-2	

Programming Notes
Cabac initialization is also dependent on the field/frame picture type, Slice type, and the current SliceQP value.

23:22 **Reserved**

MFX_AVC_SLICE_STATE

		Format: MBZ
	21:16	**Slice Quantization Parameter** Quantization Parameter for current slice. Derived from PPS and slice_delta_qp syntax element in Slice Header. It is needed for CABAC context initialization and deblocking filter control. And it is also used as the starting QP value in the very first MB of a slice. It is in the range of unsigned integer 0 to 51, for 8-bit pixel bit-depth.
	15:12	**Reserved** Format: MBZ
	11:8	**Slice Beta Offset Div2** Format: S3 2's Complement Range: [-6, 6] Inclusive Specifies the offset used in accessing the deblocking filter strength tables.
	7:4	**Reserved** Format: MBZ
	3:0	**Slice Alpha C0 Offset Div2** Format: S3 2's Complement Range: [-6, 6] Inclusive Specifies the offset used in accessing the deblocking filter strength tables.
4	31:24	**Slice Vertical Position** This field specifies the position in y-direction of the first macroblock in the Slice in unit of macroblocks. The fields (Slice_MB_Start_Hor_Pos, Slice_MB_Start_Vert_Pos) are valid in VLD (decoding) mode only. They are ignored by hardware in decoding IT mode and encoding mode (whereas the position is provided by the per-macroblock object command). Derived **Programming Notes** Error Handling: Driver needs to check if FirstMbY starts at 0 on the first slice of frame. If not, driver needs to add a phantom slice with FirstMbX and FirstMbY set to 0.
	23:16	**Slice Horizontal Position** This field specifies the position in x-direction of the first macroblock in the Slice in unit of macroblocks. Derived **Programming Notes** Error Handling: Driver needs to check if FirstMbY starts at 0 on the first slice of frame. If not, driver needs to add a phantom slice with FirstMbX and FirstMbY set to 0.
	15	**Reserved** Format: MBZ
	14:0	**Slice Start Mb Num** Exists If: Decoder Only The MB number (linear MB address in a picture) at the start of a Slice, it must match with the Slice

MFX_AVC_SLICE_STATE

		Horizontal Position (Slice_MB_Start_Hor_Pos) and Vertical Position (Slice_MB_Start_Vert_Pos) in the picture. **Programming Notes** In creating the Phantom Slice for error concealment, this field should set to the total number of MB in the current picture + 1.
5	31:24	**Reserved** Format: MBZ
	23:16	**Next Slice Vertical Position** This field specifies the position in y-direction of the first macroblock in the next Slice in unit of macroblocks. This field is primarily used for error concealment. In the case that current slice is the last slice, this field should set to the height of picture (since y-direction is zero-based numbering).
	15:8	**Reserved** Format: MBZ
	7:0	**Next Slice Horizontal Position** This field specifies the position in x-direction of the first macroblock in the next Slice in unit of macroblocks. This field is primarily used for error concealment. In the case that current slice is the last slice, this field should set to 0.
6 Encoder Only	31	**Rate Control Counter Enable** To enable the accumulation of bit allocation for rate control This field enables hardware Rate Control logic. The rest of the RC control fields are only valid when this field is set to 1. Otherwise, hardware ignores these fields. **Value** / **Name** 0 – Disable 1 – Enable
	30	**ResetRateControlCounter** To reset the bit allocation accumulation counter to 0 to restart the rate control. **Value** / **Name** 0 – Not Reset 1 – Reset
	29:28	**RC Triggle Mode** **Value** / **Name** / **Description** 00b – Always Rate Control – Whereas RC becomes active if sum_act > sum_target or sum_act < sum_target 01b – Gentle Rate Control – whereas RC becomes active if sum_act > upper_midpt or sum_act < lower_midpt 10b – Loose Rate Control – whereas RC becomes active if sum_act > sum_max or sum_act < sum_min 11b – Reserved
	27:24	**RC Stable Tolerance** Format: U4 This field specifies the tolerance required to deactivate RC once it has been triggered. **Value** / **Name** 0-15
	23	**RC Panic Enable** If this field is set to 1, RC enters panic mode when sum_act > sum_max. RC Panic Type field controls what type of panic behavior is invoked. **Value** / **Name**

0	Disable
1	Enable

22 **RC Panic Type**

This field selects between two RC Panic methods

Value	Name
0	QP Panic
1	CBP Panic

Programming Notes
If it is set to 0, in panic mode, the macroblock QP is maxed out, setting to requested QP + QP_max_pos_mod. If it is set to 1, for an intra macroblock, AC CBPs are set to zero (note that DC CBPs are not modified). For inter macroblocks, AC and DC CBPs are forced to zero.

21 **MB Type Direct Conversion Disable**

Exists If:	B-Slice

For all Macroblock type conversions in different slices, refer to Section "Macroblock Type Conversion Rules" in the same volume.

Value	Name
0	Enable direct mode conversion
1	Disable direct mode conversion

Programming Notes
This field is zero for all other slices other than B-Slice.

20 **MB Type Skip Conversion Disable**

Exists If:	P-Slice or B-Slice

For all Macroblock type conversions in different slices, refer to Section "Macroblock Type Conversion Rules" in the same volume.

Value	Name
0	Enable skip type conversion
1	Disable skip type conversion

Programming Notes
This field is zero for all other slices other than P_Slice or B-Slice. \

19 **Is Last Slice**

It is used by the zero filling in the Minimum Frame Size test.

Value	Name	Description
1		Current slice is the last slice of a picture
0		Current slice is NOT the last slice of a picture

17 **Header Insertion Present in Bitstream**

Value	Name	Description
0		No header insertion into the output bitstream buffer, in front of the current slice encoded bits.
1		Header insertion into the output bitstream buffer is present, and is in front of the current slice encoded bits.

16 **SliceData Insertion Present in Bitstream**

MFX_AVC_SLICE_STATE

		Value / **Name** / **Description** 0 / / No Slice Data insertion into the output bitstream buffer 1 / / Slice Data insertion into the output bitstream buffer is present.
	15	**Tail Insertion Present in bitstream** **Value** / **Name** / **Description** 0 / / No tail insertion into the output bitstream buffer, after the current slice encoded bits 1 / / Tail insertion into the output bitstream buffer is present, and is after the current slice encoded bits.
	14	**Reserved** Format: MBZ
	13	**EmulationByteSliceInsertEnable** To have PAK outputting SODB or EBSP to the output bitstream buffer **Value** / **Name** / **Description** 0 / / outputting RBSP 1 / / outputting EBSP
	12	**CabacZeroWordInsertionEnable** To pad the end of a SliceLayer RBSP to meet the encoded size requirement. **Value** / **Name** / **Description** 0 / / No Cabac_Zero_Word Insertion 1 / / Allow internal Cabac_Zero_Word generation and append to the end of RBSP (effectively can be used as an indicator for last slice of a picture, if the assumption is only the last slice of a picture needs to insert CABAC_ZERO_WORDs.
	11:8	**Reserved** Format: MBZ
	7:4	**Slice ID [3:0]** To identify the output data (coding information record) returned for rate control from PAK to ENC and VPP.
	3:2	**Reserved** Format: MBZ
	1:0	**Stream ID [1:0]** To identify the output data (coding information record) returned for rate control from PAK to ENC and VPP.
7	31:29	**Reserved** Format: MBZ
Encoder Only	28:0	**Indirect PAK-BSE Data Start Address (Write)** Exists If: AVC Encode Mode This field specifies the memory starting address (offset) to write out the compressed bitstream data from the BSE processing. This pointer is relative to the MFC Indirect PAK-BSE Object Base Address. It is a byte-aligned address for the AVC bitstream data in both CABAC/CAVLC Modes. For Write, there is no need to have a data length field. It is assumed the global memory bound check specified in the IND_OBJ_BASE_ADDRESS command (Indirect PAK-BSE Object Access Upper Bound) will take care of any illegal write access. **Value** / **Name** 0 - 512MB /
8	31:24	**Magnitude of QP Max Negative Modifier** Format: U8 This field specifies the lower limit of the QP modifier. **Value** / **Name**
Encoder Only		

MFX_AVC_SLICE_STATE

		0-51	
	23:16	**Magnitude of QP Max Positive Modifier**	
		Format:	U8
		This field specifies the upper limit of the QP modifier.	
		Value	**Name**
		0 - 15	
	15:12	**Shrink Param - Shrink Resistance**	
		Format:	U4
		This field specifies the additional points added each time decreased correction is invoked.	
		Value	**Name**
		0 - 15	
	11:8	**Shrink Param – Shrink Init**	
		Format:	U4
		This field specifies the initial points required to trip decreased control.	
		Value	**Name**
		0 - 15	
	7:4	**Grow Param – Grow Resistance**	
		Format:	U4
		This field specifies the additional points added each time increased correction is invoked.	
		Value	**Name**
		0 - 15	
	3:0	**Grow Param – Grow Init**	
		Format:	U4
		This field specifies the initial points required to trip increased control.	
		Value	**Name**
		0 - 15	
9	31:24	**Reserved**	
		Format:	MBZ
Encoder Only	23:20	**Correct 6**	
		Format:	U4
		This field specifies the points used in the lowermost RC region when sum_act <= sum_min.	
		Value	**Name**
		0 - 15	
	19:16	**Correct 5**	
		Format:	U4
		This field specifies the points used in the fifth RC region when sum_act > sum_min but <= lower_midpt.	
		Value	**Name**
		0 - 15	
	15:12	**Correct 4**	
		Format:	U4
		This field specifies the points used in the fourth RC region when sum_act > lower_midpt but <= sum_target.	
		Value	**Name**
		0 - 15	
	11:8	**Correct 3**	
		Format:	U4
		This field specifies the points used in the third RC region when sum_act > sum_target but <=	

upper_midpt.

Value	Name
0 - 15	

Correct 2

Format:	U4

This field specifies the points used in the second RC region when sum_act > upper_midpt but <= sum_max.

Value	Name
0 - 15	

Correct 1

Format:	U4

This field specifies the points used in the topmost RC region when sum_act > sum_max.

Value	Name
0 - 15	

10 Encoder Only

ClampValues – CV7 (31:28)

CV6 (27:24)

CV5 (23:20)

CV4 (19:16)

CV3 (15:12)

CV2 (11:8)

CV1 (7:4)

CV0 - Clamp Value 0 (3:0)

Format:	U4

If the magnitude of coefficients at locations assigned with CV0 (mapping shown below) exceeds $2^{CV0}-1$, they are replaced with $2^{CV0}-1$. For coefficients at locations marked as 'none', no clamping is performed. The following mappings are only applied to luma and chroma blocks\subblocks containing AC coefficiencts (blocks\sublocks with only DC coeffs will not be clamped).

For 4x4 frame block, each coefficient is mapped to one of the eight CV values as following:

none	CV7	CV5	CV4
CV7	CV6	CV4	CV3
CV5	CV4	CV2	CV1
CV4	CV3	CV1	CV0

For 8x8 frame block, each coefficient is mapped to one of the eight CV values as following:

none	none	CV7	CV6	CV5	CV4	CV3	CV3
none	CV7	CV6	CV5	CV4	CV3	CV3	CV2
CV7	CV6	CV5	CV4	CV3	CV3	CV2	CV2
CV6	CV5	CV4	CV3	CV3	CV2	CV2	CV1
CV5	CV4	CV3	CV3	CV2	CV2	CV1	CV1
CV4	CV3	CV3	CV2	CV2	CV1	CV1	CV0
CV3	CV3	CV2	CV2	CV1	CV1	CV0	CV0
CV3	CV2	CV2	CV1	CV1	CV0	CV0	CV0

For 4x4 field block, each coefficient is mapped to one of the eight CV values as following:

none	CV6	CV3	CV1
CV7	CV6	CV3	CV1

CV5	CV4	CV2	CV0
CV5	CV4	CV2	CV0

For 8x8 field block, each coefficient is mapped to one of the eight CV values as following:

none	none	CV6	CV5	CV4	CV3	CV2	CV1
none	CV7	CV6	CV5	CV4	CV3	CV2	CV1
CV7	CV6	CV5	CV4	CV3	CV3	CV2	CV1
CV7	CV6	CV5	CV4	CV3	CV2	CV2	CV1
CV6	CV5	CV4	CV4	CV3	CV2	CV1	CV0
CV6	CV5	CV4	CV3	CV2	CV2	CV1	CV0
CV5	CV5	CV4	CV3	CV2	CV1	CV1	CV0
CV5	CV5	CV4	CV3	CV2	CV1	CV1	CV0

Value	Name
0 - 15	

2.1.4 MFX_AVC_REF_IDX_STATE Command

MFX_AVC_REF_IDX_STATE

Source:	VideoCS
Length Bias:	2

This is a slice level command and can be issued multiple times within a picture that is comprised of multiple slices. The same command is used for AVC encoder (PAK mode) and decoder (VLD mode); it is not need in decoder IT mode.

The inline data of this command is interpreted differently for encoder as for decoder. For decoder, it is interpreted as RefIdx List L0/L1 as in AVC spec., and it matches with the DXVA2 AVC API data structure for decoder in VLD mode : RefPicList[2][32] (L0:L1, 0:31 RefPic). But for encoder, it is interpreted as a Reference Index Mapping Table for L0 and L1 reference pictures. For packing the bits at the output of PAK, the syntax elements must follow the definition of RefIdxL0/L1 list according to the AVC spec. However, the decoder pipeline was designed to use a variation of that standard definition, as such a conversion (mapping) is needed to support the hardware design.

The Reference lists are needed in processing both P and B slice in AVC codec. For P-MB, only L0 list is used; for B-MB both L0 and L1 lists are needed. For a B-MB that is coded in L1-only Prediction, only L1 list is used.

Programming Notes

DXVA2 specifies that an application will create the RefPicList L0 and L1 and pass onto the driver. The content of each entry of RefPicList L0/L1[ ] is a 7-bit picture index. This picture index is the same as that of RefFrameList[ ] content. This picture index, however, is not defined the same as the frame store ID (0 to 16, 5-bits) we have implemented in H/W. Hence, driver is required to manage a table to convert between DXVA2 picture index and intel frame store ID. As such, the final RefPicList L0/L1[ ] that the driver passes onto the H/W is not the same as that defined in the DXVA2.

DWord	Bit	Description	
0	31:29	**Command Type**	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	**Pipeline**	
		Default Value:	2h MFX_AVC_REF_IDX_STATE
		Format:	OpCode
	26:24	**Command Opcode**	
		Default Value:	1h AVC
		Format:	OpCode
	23:21	**SubOpcodeA**	
		Default Value:	0h MFX_AVC_REF_IDX_STATE
		Format:	OpCode
	20:16	**SubOpcodeB**	
		Default Value:	4h MFX_AVC_REF_IDX_STATE
		Format:	OpCode
	15:12	**Reserved**	
		Format:	MBZ
	11:0	**DWord Length**	
		Default Value:	0008h
		Excludes DWords 0,1	
1	31:1	**Reserved**	
		Format:	MBZ

MFX_AVC_REF_IDX_STATE

	0	**RefPicList Select** Num_ref_idx_l1_active is resulted from the specifications in both PPS and Slice Header for the current slice. However, since the full reference list L0 and/or L1 are always sent, only present flags are specified instead. This parameter is specified for Intel interface only, not present in the DXVA. **Value** / **Name** / **Description** 0 / RefPicList 0 / The list that followed represents RefList L0 (Decoder VLD mode) or Ref Idx Mapping Table L0 (Encoder PAK mode) 1 / RefPicList1 / The list that followed represents RefList L1 (Decoder VLD mode) or Ref Idx Mapping Table L1 (Encoder PAK mode)
2..9	31:0	**Reference List Entry** This set of fields is always present whenever this command is issued. It always specifies the full 32 reference pictures in the selected list, regardless they are “existing picture” or not. If a picture is non-existing, the corresponding entry should be set to all ones. Each list entry is 1 byte. A 32-bit DW can hold 4 list entries in the following format 31:24 entry X+3 (e.g. listY_3) 23:16 entry X+2 (e.g. listY_2) 15:8 entry X+1 (e.g. listY_1) 7:0 entry X (e.g. listY_0) X is replaced by the paddr[2:0] * 4 ; paddr[5:0] with 0x20 and 0x27, and Y is replaced by 0 or 1. The byte definition for a reference picture : Bit 7 : Non-Existing – indicates that frame store index that should have been at this entry did not exist and was replaced by an index 0 (a valid entry) for error concealment Bit 6 : Long term bit – set this reference picture to be used as long term reference Bit 5 : Field picture flag – indicates frame/field Bit 4:0 : Frame store index or Frame Store ID (Bit 4:1 is used to form the binding table index in intel implementation) This is the final Reference List L0 or L1 after any reordering specified in the Slice Header as well as modified by the driver, and its indices values are all translated to the intel specification. If the reference picture is a frame (Bit5 = 1), frame store ID is always an even number.

MFX_AVC_REF_IDX_STATE		
		This list is used in outputting MV information by the BSD unit in VLD mode. DMV access also reads and writes Mvlist0 using this frame store ID. If this set of fields is interpreted as Reference Index Mapping Table L0/L1, the same field alignment is followed, i.e. 4 mapping entries per DW. Each mapping entry is one byte in size, but only the least significant 5 bits [4:0] is relevant. Driver should zero all the upper bits [7:5] for each entry.

2.1.5 MFX_AVC_WEIGHTOFFSET_STATE Command

MFX_AVC_WEIGHTOFFSET_STATE	
Source:	VideoCS
Length Bias:	2

This is a slice level command and can be issued multiple times within a picture that is comprised of multiple slices. The same command is used for AVC encoder (PAK mode) and decoder (VLD and IT modes). However, since for AVC decoder VLD and IT modes, and AVC encoder mode, the implicit weights are computed in hardware, this command is not issued. For encoder, regardless of the type of weight calculation is active for the current slice (default, implicit or explicit), they are all sent to the PAK as if they were all in explicit mode. However, for implicit weight and offset, each entry contains only a 16-bit weight and no offset (offset = 0 always in implicit mode and can be hard-coded inside the hardware).The weights (and offsets) are needed in processing both P and B slice in AVC codec. For P-MB, at most only L0 list is used; for B-MB both L0 and L1 lists may be needed. For a B-MB that is coded in L1-only Prediction, only L1 list is sent.The content of this command matches with the DXVA2 AVC API data structure for explicit prediction mode only : Weights[2][32][3][2] (L0:L1, 0:31 RefPic, Y:Cb:Cr, W:0)

DWord	Bit	Description		
0	31:29	**Command Type**		
		Default Value:	3h PARALLEL_VIDEO_PIPE	
		Format:	OpCode	
	28:27	**Pipeline**		
		Default Value:	2h MFX_ AVC_ WEIGHTOFFSET_STATE	
		Format:	OpCode	
	26:24	**Media Command Opcode**		
		Default Value:	1h AVC_COMMON	
		Format:	OpCode	
	23:21	**SubOpcode A**		
		Default Value:	0h	
		Format:	OpCode	
	20:16	**SubOpcode B**		
		Default Value:	5h	
		Format:	OpCode	
	15:12	**Reserved**		
		Project:	All	
		Format:	MBZ	
	11:0	**DWord Length**		
		Default Value:	0h Excludes DWord (0,1) = 0030h	
		Project:	All	
		Format:	=n Total Length - 2	

MFX_AVC_WEIGHTOFFSET_STATE

1	31:1	**Reserved** Project: All Format: MBZ
	0	**Weight and Offset Select** It must be set in consistent with the WeightedPredFlag and WeightedBiPredIdc in the Img_State command. This parameter is specified for Intel interface only, not present in the DXVA. For implicit even though only one entry may be used, still loading the whole 32-entry table. **Value** / **Name** / **Description** / **Project** 0 / Weight and Offset L0 table / The list that followed is associated with the weight and offset for RefPicList L0 / All 1 / Weight and Offset L1 table / The list that followed is associated with the weight and offset for RefPicList L1 / All
2..97	31:0	**WeightOffset** WeightOffset[L=L0=0 or L1=1][i=0 to 31][Y=0/Cb=1/Cr=2][weight=0/offset=1] WeightOffset[L][i=0][Y=0][Weight=0], WeightOffset[L][i=0][Y=0][Offset=1] WeightOffset[L][i=0][Cb=1][Weight=0], WeightOffset[L][i=0][Cb=1][Offset=1] WeightOffset[L][i=0][Cr=2][Weight=0], WeightOffset[L][i=0][Cr=2][Offset=1]: WeightOffset[L][i=31][Y=0][Weight=0], WeightOffset[L][i=31][Y=0][Offset=1] WeightOffset[L][i=31][Cb=1][Weight=0], WeightOffset[L][i=31][Cb=1][Offset=1] WeightOffset[L][i=31][Cr=2][Weight=0], WeightOffset[L][i=31][Cr=2][Offset=1] Format for explicit: Both Weight and Offset are S15 in two's compliment, with a valid range from -128 to 128 Format for implicit: S15 This set of fields is always present whenever this command is issued. The full table, one entry for each reference picture, is always specified. Any reference list L0/L1[i] that does not exist, the corresponding weight and offset are set to 0. Weight and Offset are 2 byte each. Apair of Weight and Offset forms a dword, with Weight in the LOWER word and Offset in the HIGHER word. WeightOffset[L0=0][i=0 to 31][Y=0] (i.e. luma_weight_l0[i]) are specified for the weighting and offset factors applied to the luma prediction value for list 0 prediction using RefPicList0[i] (one-to-one correspondence in i). When luma_weight_l0_flag (Slice Header syntax element) is equal to 1, the value of luma_weight_l0[i] shall be in the range of –128 to 127. When luma_weight_l0_flag is equal to 0, luma_weight_l0[i] shall be inferred to be equal to 2luma_log2_weight_denom for RefPicList0[i]. luma_log2_weight_denom is a Slice Header syntax element. WeightOffset[L0=0][i=0 to 31][Cb=1] (i.e. chromaCb_weight_l0[i]) are specified for the weighting and offset factors applied to the chroma Cb prediction values for list 0 prediction using RefPicList0[i] (one-to-one correspondence in i). When chroma_weight_l0_flag (Slice Header syntax element) is equal to 1, the value of chromaCb_weight_l0[i] shall be in the range of –128 to 127. When chroma_weight_l0_flag is equal to 0, chromaCb_weight_l0[i] shall be inferred to be equal to 2chroma_log2_weight_denom for RefPicList0[i]. chroma_log2_weight_denom is a Slice Header syntax element. WeightOffset[L0=0][i=0 to 31][Cr=2] (i.e. chromaCr_weight_l0[i]) are specified for the weighting and offset factors applied to the chroma Cr prediction values for list 0 prediction using RefPicList0[i] (one-to-one correspondence in i). When chroma_weight_l0_flag (Slice Header syntax element) is equal to 1,

MFX_AVC_WEIGHTOFFSET_STATE		
		the value of chromaCr_weight_l0[i] shall be in the range of –128 to 127. When chroma_weight_l0_flag is equal to 0, chromaCr_weight_l0[i] shall be inferred to be equal to 2chroma_log2_weight_denom for RefPicList0[i].

2.2 AVC Decoder Commands

2.2.1 MFD_AVC_DPB_STATE Command

MFD_AVC_DPB_STATE		
Project:		All
Source:		VideoCS
Length Bias:		2
This is a frame level state command used only in DXVA2 AVC Short Slice Bitstream Format VLD mode. RefFrameList[16] of DXVA2 interface is replaced with intel Reference Picture Addresses[16] of MFX_PIPE_BUF_ADDR_STATE command. The LongTerm Picture flag indicator of all reference pictures are collected into LongTermPic_Flag[16]. FieldOrderCntList[16][2] and CurrFieldOrderCnt[2] of DXVA2 interface are replaced with intel POCList[34] of MFX_AVC_DIRECTMODE_STATE command.		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_MULTI_DW Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h AVC_DEC Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h Format: OpCode
	20:16	**SubOpcode B** Default Value: 6h Format: OpCode
	15:12	**Reserved** Project: All Format: MBZ
	11:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) =0009h Project: All Format: =n Total Length - 2
1	31:16	**LongTermFrame_Flag[16][1 bit]**

MFD_AVC_DPB_STATE

One-to-one correspondence with the entries of the Intel RefFrameList[16]. 1 bit per reference frame.

Value	Name	Project
1	the picture is a long term reference picture	
0	the picture is a short term reference picture	

15:0 **Non-ExistingFrame_Flag[16][1 bit]**

One-to-one correspondence with the entries of the Intel RefFrameList[16]. 1 bit per reference frame.

Value	Name	Project
1	the reference picture in that entry of RefFrameList[] does not exist anymore.	
0	the reference picture in that entry of RefFrameList[] is a valid reference	

Programming Notes

When an element of the list of frames is not relevant (e.g., due to the corresponding reference entry being empty or being marked as "not used for reference"), the value of the corresponding bit of NonExistingFrameFlags shall be set to 0.

2 31:0 **UsedForReference_Flag[16][2 bits]**

One-to-one correspondence with the entries of the Intel RefFrameList[16]. 2 bits per reference frame.

Value	Name
0	indicates a frame is “not used for reference”.
1	bit[0] indicates that the top field of a frame is marked as "used for reference".
2	bit[1] indicates that the bottom field of a frame is marked as "used for reference".
3	bit[1:0] indicates that a frame (or field pair) is marked as “used for reference”.

3..10 31:0 **LTSTFrameNumList[16][16 bits]**

One-to-one correspondence with the entries of the Intel RefFrameList[16]. 16 bits per reference frame.Depending on the corresponding LongTermFrame_Flag[], the content of this field is interpreted differently.

Value	Name	Description
1	LongTermFrame_Flag[i]	LTSTFrameNumList[i] represent LongTermFrameIdx.
0	LongTermFrame_Flag[i]	LTSTFrameNumList[i]represent Short Term Picture FrameNum.

Programming Notes

When an element of the list of frames is not relevant (e.g., due to the corresponding reference entry being empty or being marked as "not used for reference"), the value of the LTSTFrameNumList entry shall be set to 0.

NOTE modified from DXVA2 – The values in RefFrameList and UsedForReference_Flag are the primary means by which the H/W can determine whether the corresponding entries in RefFrameList, POCList, LTSTFrameNumList, and Non-ExistingFrame_Flag should be considered valid for use in the decoding process of the current picture or not. When RefFrameList[i] is marked to be invalid, the values of POCList[i][0], POCList[i][1], LTSTFrameNumList[i], UsedForReference_Flag[i], and Non-ExistingFrame_Flag[i] must all be equal to 0. When UsedForReference_Flag[i] = 0, the value of RefFrameList[i] must be marked invalid.

2.2.2 MFD_AVC_SLICEADDR Command

MFD_AVC_SLICEADDR	
Project:	All
Source:	VideoCS
Length Bias:	2

This is a Slice level command used only for DXVA2 AVC Short Slice Bitstream Format VLD mode.When decoding a slice, H/W needs to know the last MB of the slice has reached in order to start decoding the next slice. It also needs to know if a slice is terminated but the last MB has not reached, error conealment should be invoked to generate those missing MBs. For AVC DXVA2 Short Format, the only way to know the last MB position of the current slice, H/W needs to snoop into the next slice's start MB address (a linear address encoded in the Slice Header). Since each BSD Object command can have only one indirect bitstream buffer address, this command is added to help H/W to snoop into the next slice's slice header and retrieve its Start MB Address. This command will take the next slice's bitstream buffer address as input (exactly the same way as a BSD Object command), and parse only the first_mb_in_slice syntax element. The result will stored inside the H/W, and will be used to decode the current slice specified in the BSD Object command.Only the very first few bytes (max 5 bytes for a max 4K picture) of the Slice Header will be decoded, the rest of the bitstream are don't care. This is because the first_mb_in_slice is encoded in Exponential Golomb, and will take 33 bits to represent the max 256 x 256 = 64K-1 value. The indirect data of MFD_AVC_SLICEADDR is a valid BSD object and is decoded as in BSD OBJECT command.The next Slice Start MB Address is also exposed to the MMIO interface.The Slice Start MB Address (first_mb_in_slice) is a linear MB address count; but it is translated into the corresponding 2D MB X and Y raster position, and are stored internally as NextSliceMbY and NextSliceMbX.

DWord	Bit	Description	
0	31:29	**Command Type**	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	**Pipeline**	
		Default Value:	2h MFD_AVC_ SLICEADDR
		Format:	OpCode
	26:24	**Media Command Opcode**	
		Default Value:	1h AVC_DEC
		Format:	OpCode
	23:21	**SubOpcode A**	
		Default Value:	1h
		Format:	OpCode
	20:16	**SubOpcode B**	
		Default Value:	7h
		Format:	OpCode
	15:12	**Reserved**	
		Project:	All
		Format:	MBZ
	11:0	**DWord Length**	
		Default Value:	0h Excludes DWord (0,1) =0001h
		Project:	All
		Format:	=n Total Length - 2
1	31:24	**Reserved**	
		Project:	All
		Format:	MBZ

MFD_AVC_SLICEADDR		
	23:0	**Indirect BSD Data Length** Format: U24 in bytes This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled – subsequently, the Indirect Data Start Address field is ignored. Driver always programs this up to 5 bytes; for bitstream less than 5 bytes, driver program the lesser value. (Emulation Prevention Byte should never happen for the first 5 bytes when the max picture size can only be 4Kx4K)It is the length in bytes of the bitstream data for the current slice, including Slice Header + Slice Data + Emulation Prevention Bytes + any filling trailing zeros after the last MB. Hardware ignores the contents after the last non-zero byte. Trailing zero is allowed and handled correctly in both CABAC and CAVLC modes.
2	31:29	**Reserved** Project: All Format: MBZ
	28:0	**Indirect BSD Data Start Address** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This pointer is relative to the MFD Indirect Object Base Address.Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the AVC bitstream data in both CABAC/CAVLD Modes.In implementing a phantom slice at the end of a picture for automatic error concealment, this field should set to 0.It includes the NAL Header Byte. (but does not perform EMU detection).Must provide a valid MB address, even if error. MB must be clamped to within a pic boundary. Value: [0,512MB) — Name:

2.2.3 MFD_AVC_BSD_OBJECT Command

MFD_AVC_BSD_OBJECT		
Source: VideoCS Length Bias: 2 The MFD_AVC_BSD_OBJECT command is the only primitive command for the AVC Decoding Pipeline. The same command is used for both CABAC and CAVLD modes. The Slice Data portion of the bitstream is loaded as indirect data object.Before issuing a MFD_AVC_BSD_OBJECT command, all AVC states of the MFD Engine need to be valid. Therefore the commands used to set these states need to have been issued prior to the issue of a MFD_AVC_BSD_OBJECT command.Context switch interrupt is not supported by this command.		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFD_AVC_BSD_OBJECT

MFD_AVC_BSD_OBJECT

		Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h AVC_DEC Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h Format: OpCode
	20:16	**SubOpcode B** Default Value: 8h Format: OpCode
	15:12	**Reserved** Project: All Format: MBZ
	11:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) = 0004 Project: All Format: =n Total Length - 2
1	31:24	**Reserved** Format: MBZ
	23:0	**Indirect BSD Data Length** Format: U24 This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled – subsequently, the Indirect Data Start Address field is ignored. This field must have the same alignment as the Indirect Object Data Start Address. AVC Short Format : It is the length in bytes of the bitstream data for the current slice, including Slice Header + Slice Data + Emulation Prevention Bytes + any filling trailing zeros after the last MB. Hardware ignores the contents after the last non-zero byte. Trailing zero is allowed and handled correctly in both CABAC and CAVLC modes.
2	31:29	**Reserved** Project: All Format: MBZ
	28:0	**Indirect BSD Data Start Address** Project: All Format: U29 This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This pointer is relative to the **MFD Indirect Object Base Address**. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the AVC bitstream data in both CABAC/CAVLD Modes. In implementing a phantom slice at the end of a picture for automatic error concealment, this field should set to 0. It includes the NAL Header (the NAL Header does not need to perform EMU detection). For AVC Base Layer, it is a single byte. But for MVC, the NAL Header is 4 Bytes long. These NAL Header Unit must be passed to HW in the compressed bitstream buffer. **Value** / **Name** [0,512MB)
3..5	31:0	**Inline Data**

MFD_AVC_BSD_OBJECT		
		All the required Slice Header parameters and error handling settings are captured as InLine Data of the AVC_BSD_OBJECT command. It has a fixed size of 4 DWs. Its definition is described in the follwoing section: Inline Data Description.

2.2.3.1 Inline Data Description

Inline Data Description

Source: VideoCS

Default Value: 0x00000000, 0x00000000

This structure includes all the required Slice Header parameters and error handling settings for AVC_BSD_OBJECT command.

DWord	Bit	Description
3	31	**Concealment Method** This field specifies the method used for concealment when error is detected. If set, a copy from collocated macroblock location is performed from the concealment reference indicated by the ConCeal_Pic_Id field. If it is not set, a copy from the current picture is performed using Intra 16x16 Prediction method. Value / Name / Description: 0 — Intra 16x16 Prediction 1 — Inter P Copy
	30	**Init Current MB Number** When set, the current Slice_Start_MB_Num, Slice_MB_Start_Hor_Pos and Slice_MB_Start_Vert_Pos fields will be used to initialize the Current_MB_Number register. This effectively disables the concealment capability.
	29	**Reserved** Format: MBZ
	28:27	**MB Error Concealment B Temporal Prediction mode** These two bits control how the reference L0/L1 are overridden in B temporal slice. Value / Name / Description: 00b — [Default] — Both Reference Indexes L0/L1 are forced to 0 during Concealment 01b — Only Reference Index L1 is forced to 0; Reference Index L0 is forced to -1 10b — Only Reference Index L0 is forced to 0; Reference Index L1 is forced to -1 11b — Reserved — Invalid
	26	**MB Error Concealment B Temporal Reference Index Override Enable Flag** During MB Error Concealment on B slice with Temporal Direct Prediction, either L0 or L1 or both can be forced to 0 (MB Error Concealment B Temporal Reference Index Override Mode from above will control which one) This bit can be set to use the predicted reference indexes instead. Value / Name / Description: 0 — [Default] — Predicted Reference Indexes L0/L1 are used during MB Concealment. 1 — Reference Indexes L0/L1 are overridden to 0 during MB Concealment.
	25	**MB Error Concealment B Temporal Motion Vectors Override Enable Flag** During MB Error Concealment on B slice with Temporal Direct Prediction, motion vectors are forced to

0 to improve image quality.
This bit can be set to preserve the original weight prediction.

Value	Name	Description
0	[Default]	Predicted Motion Vectors are used during MB Concealment
1		Motion Vectors are Overridden to 0 during MB Concealment

24 **MB Error Concealment B Temporal Weight Prediction Disable Flag**

During MB Error Concealment on B slice with Temporal Direct Prediction, weight prediction is disabled to improve image quality.
This bit can be set to preserve the original weight prediction.

Value	Name	Description
0	[Default]	Weight Prediction is Disabled during MB Concealment
1		Weight Prediction will not be overridden during MB Concealment

23:22 **Reserved**

Format:	MBZ

21:16 **Concealment Picture ID**

This field identifies the picture in the reference list to be used for concealment. This field is only valid if **Concealment Method** is Inter P Copy.

Bit Filed	Value	Defenition
21	0	Frame Picture
21	1	Field picture
20:16	All	Frame Store Index[4:0]

15 **Reserved**

Format:	MBZ

14 **BSD Premature Complete Error Handling**

BSD Premature Complete Error occurs in situation where the Slice decode is completed but there are still data in the bitstream.

Value	Name	Description
1		Set the interrupt to the driver (provide MMIO registers for MB address R/W)
0		Ignore the error and continue (masked the interrupt), assume the hardware automatically performs the error handling

13 **Reserved**

Format:	MBZ

12 **MPR Error (MV out of range) Handling**

Software must follow the action for each Value as follow:

Value	Name	Description
1		Set the interrupt to the driver (provide MMIO registers for MB address R/W)
0		Ignore the error and continue (masked the interrupt), assume the hardware automatically performs the error handling

11 **Reserved**

Format:	MBZ

10 **Entropy Error Handling**

Software must follow the action for each Value as follow:

Value	Name	Description
1		Set the interrupt to the driver (provide MMIO registers for MB address R/W).
0		Ignore the error and continue (masked the interrupt), assume the hardware automatically perform the error handling.

9 **Reserved**

Format:	MBZ

Inline Data Description

8 — MB Header Error Handling

Software must follow the action for each Value as follow:

Value	Name	Description
1		Set the interrupt to the driver (provide MMIO registers for MB address R/W).
0		Ignore the error and continue (masked the interrupt), assume the hardware automatically perform the error concealment.

7:6 — MB Error Concealment B Spatial Prediction mode

These two bits control how the reference L0/L1 are overridden in B spatial slice.

Value	Name	Description
00b	**[Default]**	Both Reference Indexes L0/L1 are forced to 0 during Concealment
01b		Only Reference Index L1 is forced to 0; Reference Index L0 is forced to -1
10b		Only Reference Index L0 is forced to 0; Reference Index L1 is forced to -1
11b	Reserved	Invalid

5 — MB Error Concealment B Spatial Reference Index Override Disable Flag

During MB Error Concealment on B slice with Spatial Direct Prediction, either L0 or L1 or both can be forced to 0 (MB Error Concealment B Spatial Reference Index Override Mode from above will control which one)

This bit can be set to use the predicted reference indexes instead.

Value	Name	Description
0	**[Default]**	Reference Indexes L0/L1 are overridden during MB Concealment
1		Predicted Reference Indexes L0/L1 are used during MB Concealment

4 — MB Error Concealment B Spatial Motion Vectors Override Disable Flag

During MB Error Concealment on B slice with Spatial Direct Prediction, motion vectors are forced to 0 to improve image quality.

This bit can be set to use the predicted motion vectors instead.

This bit does not affect normal decoded MB.

Value	Name	Description
0	**[Default]**	Motion Vectors are Overridden to 0 during MB Concealment
1		Predicted Motion Vectors are used during MB Concealment

3 — MB Error Concealment B Spatial Weight Prediction Disable Flag

During MB Error Concealment on B slice with Spatial Direct Prediction, weight prediction is disabled to improve image quality.

This bit can be set to preserve the original weight prediction.

This bit does not affect normal decoded MB.

Value	Name	Description
0	**[Default]**	Weight Prediction is Disabled during MB Concealment.
1		Weight Prediction will not be overridden during MB Concealment.

2 — MB Error Concealment P Slice Reference Index Override Disable Flag

During MB Error Concealment on P slice reference index L0 is forced to 0.

This bit can be set to use the predicted reference indexes instead.

This bit does not affect normal decoded MB.

Value	Name	Description
0	**[Default]**	Reference Indexes L0 are force to 0
1		Predicted Reference Indexes L0 are used during MB Concealment.

1 — MB Error Concealment P Slice Motion Vectors Override Disable Flag

During MB Error Concealment on P slice, motion vectors are forced to 0 to improve image quality.

This bit can be set to use the predicted motion vectors instead.

Inline Data Description

This bit does not affect normal decoded MB.

Value	Name	Description
0	**[Default]**	Motion Vectors are Overridden to 0 during MB Concealment
1		Predicted Motion Vectors are used during MB Concealment

0 — MB Error Concealment P Slice Weight Prediction Disable Flag

During MB Error Concealment on P slice, weight prediction is disabled to improve image quality.
This bit can be set to preserve the original weight prediction.
This bit does not affect normal decoded MB.

Value	Name	Description
0	**[Default]**	Weight Prediction is Disabled during MB Concealment.
1		Weight Prediction will not be overridden during MB Concealment.

4 — 31:16 First MB Byte Offset of Slice Data or Slice Header

Description	Project
Long Format: It gives the byte offset to locate the Slice Header in the bitstream for a slice, provided by the Indirect BSD Data Start Address. It does not include any Emulation Byte count present in the Slice Header. HW will take care of the Emulation Byte adjustment to this offset.	
Short Format: it should be programmed to be 0. HW will parse the Slice Header.	

Programming Notes	Project
MFX supports only DXVA2 Long and Short Format.	

15:8 Reserved

Format:	MBZ

7 — Fix Prev Mb Skipped

Enables an alternative method for decoding mb_skipped, to cope with an encoder that codes a skipped MB as a direct MB with no coefficient.

6:5 Reserved

Format:	MBZ

Programming Notes
Please note that the field MUST be set to '0' at this time.

4 — Emulation Prevention Byte Present

Value	Name	Description
0		H/W needs to perform Emulation Byte Removal
1		H/W does not need to perform Emulation Byte Removal

3 — LastSlice Flag

It is needed for both error concealment at the end of a picture. It is also needed to know to set the last MB in a picture correctly.

Value	Name	Description
1		If the current Slice to be decoded is the very last slice of the current picture.
0		If the current Slice to be decoded is any slice other than the very last slice of the current picture

2:0 First Macroblock (MB)Bit Offset

Exists If:	AVC Long Format Only
Format:	U3

This field provides the bit offset of the first macroblock of the Slice in the first byte of the input compressed bitstream.

Inline Data Description		

2.3 AVC Encoder PAK Commands

Each PAK Commands is composed of a command op-code DW and one or more command data DWs (inline data). The size of each command is specified as part of the op-code DW. Most of the commands have fixed size, except some are allowed to be of variable length.

There is an inherent order of executing MFC PAK commands that driver must follow.

2.3.1 MFC_AVC_PAK_OBJECT Command

MFC_AVC_PAK_OBJECT	
Source:	VideoCS
Length Bias:	2

The MFC_AVC_PAK_OBJECT command is the second primitive command for the AVC Encoding Pipeline. The same command is used for both CABAC and CAVLC modes. The MV Data portion of the bitstream is loaded as indirect data object.Before issuing a MFC_AVC_PAK_OBJECT command, all AVC MFX states need to be valid. Therefore the commands used to set these states need to have been issued prior to the issue of this command.MB record must be consecutive with no gaps, hence we do not need MB(x,y) in each MB command. Internal counter will keep track of the current MB address, starting from the Start_MB_In_Slice loaded at the beginning of each slice. MFC_AVC_PAK_OBJECT command follows the MbType definition like MFD. Many fields in this command are identical to that in VME output. This is intended to reduce software converting overhead from VME to PAK. Encoding statistical data such as the total size of the output bitstream are provided through MMIO registers. Software may access these registers through MI_STORE_REGISTER_MEM command.

DWord	Bit	Description	
0	31:29	**Command Type**	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	**Pipeline**	
		Default Value:	2h MFC_AVC_PAK_OBJECT
		Format:	OpCode
	26:24	**Media Command Opcode**	
		Default Value:	1h AVC_ENC
		Format:	OpCode
	23:21	**SubOpcode A**	
		Default Value:	2h
		Format:	OpCode
	20:16	**SubOpcode B**	
		Default Value:	9h
		Format:	OpCode
	15:12	**Reserved**	
		Project:	All
		Format:	MBZ

MFC_AVC_PAK_OBJECT		
	11:0	**DWord Length** Default Value: 000Ah DWORD_COUNT_n Project: All Format: =n Length -2
1	31:10	**Reserved** Project: All Format: MBZ
	9:0	**Indirect PAK-MV Data Length** Format: U10 This field provides the length in bytes of the indirect data, which contains all the MVs for the current MB (in any partitioning and subpartitioning form). A value zero indicates that indirect data fetching is disabled – subsequently, the Indirect PAK-MV Data Start Address field is ignored. This field must have the same alignment as the Indirect PAK-MV Data Start Address. This field must be DW aligned (since each MV is 4 bytes in size).Driver has to derived this field from MVsize (MVquantity in DXVA, exact size) *4 bytes per MV.
2	31:29	**Reserved** Format: MBZ
	28:0	**Indirect PAK-MV Data Start Address Offset.** This field specifies the memory starting address (offset) of the MV data to be fetched into PAK Subsystem for processing. This pointer is relative to the MFC Indirect PAK-MV Object Base Address.Hardware ignores this field if indirect data is not present, i.e. the Indirect PAK-MV Data Length is set to 0. It is a Dword aligned address in all AVC encoding configuration, since each MV is 4 bytes in size. Value: [0,512MB) — Name:
3..10	31:0	**Inline Data** All the required MB level controls and parameters for encoding are captured as inline data of the MFC_AVC_PAK_OBJECT command. It has a fixed size of 8 DWs. Its definition is described in the next section.

2.3.1.1 PAK Object Inline Data Description

The Inline Data includes all the required MB encoding states, constitute part of the Slice Data syntax elements, MB Header syntax elements and their derivatives. It provides information for the following operations:

1. Forward and Inverse Transform
2. Forward and Inverse Quantization
3. Advanced Rate Control (QRC)
4. MB Parameter Construction (MPC)
5. CABAC/CAVLC encoding
6. Bit stream packing
7. Intra and inter-Prediction decoding loop
8. Internal error handling

These state/parameter values may subject to change on a per-MB basis, and must be provided in each MFC_AVC_PAK_OBJECT command. The values set for these variables are retained internally, until they are reset by hardware Asynchronous Reset or changed by the next MFC_AVC_PAK_OBJECT command.

The inline data has been designed to match the DXVA 2.0, with the exception of the starting byte (DW0:0-7) and the ending dword (DW7:0-31).

The Deblocker Filter Control flags (FilterInternalEdgesFlag, FilterTopMbEdgeFlag and FilterLeftMbEdgesFlag) are generated by H/W, which are depending on MbaffFrameFlag, CurrMbAddr, PicWidthInMbs and disable_deblocking_filter_idc states.

Current MB [x,y] address is not sent, it is assumed that the H/W will keep track of the MB count and current MB position internally.

DWord	Bit	Description
3	31	**ExtendedForm** This field specifies that **LumaIntraMode** and **RefPicSelect** are fully replicated in 4x4 and 8x8 sub-blocks respectively. This non-DXVA form is used for optimal kernel performance.
	30:24	**Reserved: MBZ**
	23	**Reserved : MBZ**
	22:20	**MvFormat (Motion Vector Size)**. This field specifies the size and format of the output motion vectors. This field is reserved (MBZ) when the **IntraMbFlag** = 1. The valid encodings are: **000 = 0: No motion vector** **100 = 8MV: Four 8x8 motion vector pairs** **110 = 32MV: 16 4x4 motion vector pairs** Others are reserved. *(The following encodings are intended for future usages:* *001 = 1MV: one 16x16 motion vector* *010 = 2MV: One 16x16 motion vector pair* *011 = 4MV: Four 8x8 motion vectors* *101 = 16MV: 16 4x4 motion vectors* *111 = Packed, number of MVs is given by* **PackedMvNum***.)* **Note:** This field is fully supported for 100 (8MV) and 110 (32MV)
	19	**CbpDcY.** This field specifies if the Luma DC sub-block is coded. Setting it to 0 will force PAK to zero out the Luma sub-block. Otherwise, whether the sub-block is coded will be determined by the quantization process. 1 – the 4x4 DC-only Luma sub-block of the Intra16x16 coded MB is present; it is still possible that all DC coefficients are zero. 0 – no 4x4 DC-only Luma sub-block is present; either not in Intra16x16 MB mode or all DC

DWord	Bit	Description
		coefficients are zero. **Programming Note:** when Reference Mb: IPCM or inferred IPCM, current mb: base mode flag = 1; all bits in CbpDcY, CbpDcU, CbpDcV, Cbp4x4Y[15:0], Cbp4x4V[15:0] and Cbp4x4U[15:0] must set to 1's.
	18	**CbpDcU.** This field specifies if the Chroma Cb DC sub-block is coded. Setting it to 0 will force PAK to zero out the Luma sub-block. Otherwise, whether the sub-block is coded will be determined by the quantization process. 1 – the 2x2 DC-only Chroma Cb sub-block of all coded MB (any type) is present; it is still possible that all DC coefficients are zero. 0 – no 2x2 DC-only Chroma Cb sub-block is present; all DC coefficients are zero. **Programming Note:** when Reference Mb: IPCM or inferred IPCM, current mb: base mode flag = 1; all bits in CbpDcY, CbpDcU, CbpDcV, Cbp4x4Y[15:0], Cbp4x4V[15:0] and Cbp4x4U[15:0] must set to 1's.
	17	**CbpDcV.** This field specifies if the Chroma Cb DC sub-block is coded**.** Setting it to 0 will force PAK to zero out the Luma sub-block. Otherwise, whether the sub-block is coded will be determined by the quantization process. 1 – the 2x2 DC-only Chroma Cr sub-block of all coded MB (any type) is present; it is still possible that all DC coefficients are zero. 0 – no 2x2 DC-only Chroma Cr sub-block is present; all DC coefficients are zero. **Programming Note:** when Reference Mb: IPCM or inferred IPCM, current mb: base mode flag = 1; all bits in CbpDcY, CbpDcU, CbpDcV, Cbp4x4Y[15:0], Cbp4x4V[15:0] and Cbp4x4U[15:0] must set to 1's.
	16	**Reserved: MBZ** (reserved for future use as **ExternalResidBufFlag** for turbo mode)
	15	**Transform8x8Flag** This field indicates that 8x8 transform is used for the macroblock. When it is set to 0, the current MB uses 4x4 transform. When it is set to 1, the current MB uses 8x8 transform. The transform_size_8x8_flag syntax element, if present in the output bitstream, is the same as this field. However, whether transform_szie_8x8_flag is present or not in the output bitstream depends on several other conditions. This field is only allowed to be set to 1 for two conditions: It must be 1 if **IntraMbFlag** = INTRA and **IntraMbMode** = INTRA_8x8 It may be 1 if **IntraMbFlag** = INTER and there is no sub partition size less than 8x8 Otherwise, this field must be set to 0. 0: 4x4 integer transform 1: 8x8 integer transform
	14	**FieldMbFlag** This field specifies the field polarity of the current macroblock, as the mb_field_decoding_flag syntax element in AVC spec. This field specifies whether current macroblock is coded as a field or frame macroblock in MBAFF mode. It is exactly the same as FIELD_PIC_FLAG syntax element in non-MBAFF

DWord	Bit	Description
		mode. 0 = Frame macroblock 1 = Field macroblock
	13	**IntraMbFlag** This field specifies whether the current macroblock is an Intra (I) macroblock. I_PCM is considered as Intra MB. For I-picture MB (IntraPicFlag =1), this field must be set to 1. This flag must be set in consistent with the interpretation of MbType (inter or intra modes). 0: INTER (inter macroblock) 1: INTRA (intra macroblock)
	12:8	**MbType5Bits** This field is encoded to match with the best macroblock mode determined as described in the next section. It follows an unified encoding for inter and intra macroblocks according to AVC Spec.
	7	**FieldMbPolarityFlag** This field indicates the field polarity of the current macroblock. Within an MbAff frame picture, this field may be different per macroblock and is set to 1 only for the second macroblock in a MbAff pair if FieldMbFlag is set. Otherwise, it is set to 0. Within a field picture, this field is set to 1 if the current picture is the bottom field picture. Otherwise, it is set to 0. It is a constant for the whole field picture. This field is reserved and MBZ for a progressive frame picture. 0 = Current macroblock is a field macroblock from the **top** field 1 = Current macroblock is a field macroblock from the **bottom** field *Programming Note: Here bits [26:24] (MbAffFieldFlag and FiedlMbPolarityFlag) match with bits [10:8] of the Media Block Read message descriptor, simplifying the programming for message generation, as when MbAffFieldFlag is "1", kernels need to override the original "frame" surface state set for MBAFF frame picture.*
	6	MB Reserved: Inter MB converted to IPCM This field is used for HW purpose only SW should not use it.
	5:4	**IntraMbMode** This field is provided to carry information partially overlapped with MbType. This field is only valid if **IntraMbFlag** = INTRA, otherwise, it is ignored by hardware..
	3	Reserved: MBZ
	2	**SkipMbFlag** By setting it to 1, this field forces an inter macroblock to be encoded as a skipped macroblock. It is equivalent to mb_skip_flag in AVS spec, indicating that a macroblock is inferred as a P_Skip (or B_Skip) in a P Slice (or B Slice). Hardware honors input MVs for

DWord	Bit	Description
		motion prediction and forces CBP to zero. By setting it to 0, an inter macroblock will be coded as a normal inter macroblock. The macroblock may still be coded as a skipped macroblock, according to the macroblock type conversion rules described in the later sub sections. This field can only be set to 1 for certain values of MbType. See details later. This field is only valid for an inter macroblock. For intra MB (bit 13 of this DW set to one), this bit must be set to zero. 0 = not a skipped macroblock 1 = is coded as a skipped macroblock
	1:0	**InterMbMode** This field is provided to carry redundant information as that encoded in MbType. This field is only valid if **IntraMbFlag** =0, otherwise, it is ignored by hardware.
4	31:16	Cbp4x4Y[bit 15:0] (Coded Block Pattern Y) For 4x4 sub-block (when Transform8x8flag = 0 or in intra16x16) : 16-bit cbp, one bit for each 4x4 Luma sub-block (not including the DC 4x4 Luma block in intra16x16) in a MB. The 4x4 Luma sub-blocks are numbered as blk0 1 4 5 bit15 14 11 10 lk2 3 6 7 bit13 12 9 8 blk8 9 12 13 bit7 6 3 2 blk10 11 14 15 bit5 4 1 0 The cbpY bit assignment is cbpY bit [15 - X] for sub-block_num X. For 8x8 block (when Transform8x8flag = 1) Only the lower 4 bits [3:0] are valid; the remaining upper bits [15:4] are ignored. The 8x8 Luma blocks are numbered as blk0 1 bit3 2 blk2 3 bit1 0 The cbpY bit assignment is cbpY bit [3 - X] for block_num X. 0 in a bit - indicates the corresponding 8x8 block or 4x4 sub-block is not present (because all coefficient values are zero), or force to zero for PAK. 1 in a bit - indicates the corresponding 8x8 block or 4x4 sub-block is present (although it is still possible to have all its coefficients be zero - bad coding). **Programming Note:** when Reference Mb: IPCM or inferred IPCM, current mb: base mode flag = 1; all bits in CbpDcY, CbpDcU, CbpDcV, Cbp4x4Y[15:0], Cbp4x4V[15:0] and Cbp4x4U[15:0] must set to 1's.
	15:8	MbYCnt (Vertical Origin). This field specifies the vertical origin of current macroblock in the destination picture in units of macroblocks.

DWord	Bit	Description
		Format = U8 in unit of macroblock.
	7:0	MbXCnt (Horizontal Origin). This field specifies the horizontal origin of current macroblock in the destination picture in units of macroblocks. Format = U8 in unit of macroblock.
5	31:16	Cbp4x4V (Coded Block Pattern Cr) Only the lower 4 bits [3:0] are valid for 4:2:0. The 4x4 Cr sub-blocks are numbered as blk0 1 bit3 2 blk2 3 bit1 0 The cbpCr bit assignment is cbpCr bit [3 - X] for sub-block_num X. 0 in a bit - indicates the corresponding 4x4 sub-block is not present (because all coefficient values are zero), or force to zero for PAK. 1 in a bit - indicates the corresponding 4x4 sub-block is present (although it is still possible to have all its coefficients be zero - bad coding). For monochrome, this field is ignored. **Programming Note:** when Reference Mb: IPCM or inferred IPCM, current mb: base mode flag = 1; all bits in CbpDcY, CbpDcU, CbpDcV, Cbp4x4Y[15:0], Cbp4x4V[15:0] and Cbp4x4U[15:0] must set to 1's.
	15:0	Cbp4x4U (Coded Block Pattern Cb) Only the lower 4 bits [3:0] are valid for 4:2:0. The 4x4 Cb sub-blocks are numbered as blk0 1 bit3 2 blk2 3 bit1 0 The cbpCb bit assignment is cbpCb bit [3 - X] for sub-block_num X. 0 in a bit - indicates the corresponding 4x4 sub-block is not present (because all coefficient values are zero), or force to zero for PAK. 1 in a bit - indicates the corresponding 4x4 sub-block is present (although it is still possible to have all its coefficients be zero - bad coding). For monochrome, this field is ignored. **Programming Note:** when Reference Mb: IPCM or inferred IPCM, current mb: base mode flag = 1; all bits in CbpDcY, CbpDcU, CbpDcV, Cbp4x4Y[15:0], Cbp4x4V[15:0] and Cbp4x4U[15:0] must set to 1's.
6	31:28	**Skip8x8Pattern** This field indicates whether each of the four 8x8 sub macroblocks is using the predicted MVs and will not be explicitly coded in the bitstream (the sub macroblock will be coded as direct mode). It contains four 1-bit subfields, corresponding to the 4 sub macroblocks in sequential order. The whole macroblock may be actually coded as B_Direct_16x16 or B_Skip, according to the macroblock type conversion rules described in a later sub section. This field is only valid for a B slice. It is ignored by hardware for a P slice. Hardware also ignores this field for an intra macroblock. 0 in a bit – Corresponding MVs are sent in the bitstream 1 in a bit – Corresponding MVs are not sent in the bitstream
	27	**EnableCoeffClamp** 1 = the magnitude of coefficients of the current MB will be clamped based on the clamping matrix after quantization 0 = no clamping
	26	**LastMbFlag**

DWord	Bit	Description
		1 – the current MB is the last MB in the current Slice 0 – the current MB is not the last MB in the current SliceReserved MBZ.
	25	**SkipMbConvDisable** This is a per-MB level control to enable and disable skip conversion. This field is ORed with SkipConvDisable field. This field is only valid for a P or B slice. It must be zero for other slice types. Rules are provided in Section *Macroblock Type Conversion Rules* 0 - Enable skip type conversion for the current macroblock 1 - Disable skip type conversion for the current macroblock
	24:8	Reserved MBZ.
	7:0	**QpPrimeY** This is the per-MB QP value specified for the current MB. For 8-bit pixel data, QpY is the same as QpPrimeY, and it takes on a value in the range of 0 to 51, positive integer. **Note:** This value may differ from the actual codes, when HW QRC is on
7 .. 9	31:0 Each	For intra macroblocks, definition of these fields are specified in *PAK Object Inline Data Description* For inter macroblocks, definition of these fields are specified in *PAK Object Inline Data Description*
10	31:24	**MaxSizeInWord** PAK should not exceed this budget accumulatively, otherwise it will trickle the PANIC mode.
	23:16	**TargetSizeInWord** PAK should use this budget accumulatively to decide if it needs to limit the number of non-zero coefficients.
	15:0	**Reserved : MBZ**

Inline data for LumaIntraMode

ExtendedForm	0 or 1	0	0	1	1
	Intra4x4	Intra8x8	Intra16x16	Intra8x8	Intra16x16
DW4 – 31:28	Block 7	-	-	-	Block 0
DW4 – 27:24	Block 6	-	-	-	Block 0
DW4 – 23:20	Block 5	-	-	-	Block 0
DW4 – 19:16	Block 4	-	-	-	Block 0
DW4– 15:12	Block 3	-	-	-	Block 0
DW4 – 11:8	Block 2	-	-	-	Block 0
DW4 – 7:4	Block 1	-	-	-	Block 0
DW4 – 3:0	Block 0	-	-	-	Block 0
DW5 – 31:28	Block 15	-	-	-	Block 0
DW5 – 27:24	Block 14	-	-	-	Block 0
DW5 – 23:20	Block 13	-	-	-	Block 0

ExtendedForm	**0 or 1**	**0**	**0**	**1**	**1**
DW5 – 19:16	Block 12	-	-	-	Block 0
DW5 – 15:12	Block 11	-	-	-	Block 0
DW5– 11:8	Block 10	-	-	-	Block 0
DW5 – 7:4	Block 9	-	-	-	Block 0
DW5 – 3:0	Block 8	-	-	-	Block 0

vctrl_pred_mode[63:0]	(vctrl_it_lumaintrapredmode3[15:0] & vctrl_it_lumaintrapredmode2[15:0] & vctrl_it_lumaintrapredmode1[15:0] & vctrl_it_lumaintrapredmode0[15:0]) : vctrl_pred_mode_noextend[63:0]
vctrl_pred_mode_noextend[63:0]	(vctrl_INTRA_vld_16x16mode & vctrl_it_Transform8x8Flag) ? vctrl_pred_mode_noextend_4x4[63:0] : vctrl_pred_mode_noextend_16x16[63:0] : vctrl_pred_mode_noextend_8x8[63:0] : vctrl_pred_mode_noextend_4x4[63:0]
vctrl_pred_mode_noextend_16x16[63:0]	vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0]
vctrl_pred_mode_noextend_8x8[63:0]	“h000” & vctrl_it_lumaintrapredmode0[15:12] & “h000” & vctrl_it_lumaintrapredmode0[11:8] & “h000” & vctrl_it_lumaintrapredmode0[7:4] & “h000” & vctrl_it_lumaintrapredmode0[3:0]
vctrl_pred_mode_noextend_4x4[63:0]	vctrl_it_lumaintrapredmode3[15:0] & vctrl_it_lumaintrapredmode2[15:0] & vctrl_it_lumaintrapredmode1[15:0] & vctrl_it_lumaintrapredmode0[15:0]

Inline data for RefPicSelect

ExtendedForm	0	0	0	0 or 1	1	1	1
	16x16	16x8	8x16	8x8	16x16	16x8	8x16
DW8 – 31:24	-	-	-	L0 blk3	L0 blk0	-	L0 blk1
DW8 – 23:16	-	-	-	L0 blk2	L0 blk0	-	L0 blk0
DW8 – 15:8	-	L0 blk1	L0 blk1	L0 blk1	L0 blk0	-	L0 blk1
DW8 – 7:0	L0 blk0	L0 blk0	L0 blk0	L0 blk0	L0 blk0	-	L0 blk0
DW9 – 31:24	-	-	-	L1 blk3	L1 blk0	-	L1 blk1
DW9 – 23:16	-	-	-	L1 blk2	L1 blk0	-	L1 blk0
DW9 – 15:8	-	L1 blk1	L1 blk1	L1 blk1	L1 blk0	-	L1 blk1
DW9 – 7:0	L1 blk0	L1 blk0	L1 blk0	L1 blk0	L1 blk0	-	L1 blk0

The inline data content of Dwords 4 to 6 is defined either for intra prediction or for inter prediction, but not both.

Inline data subfields for an Intra Macroblock

Dword	Bit	Description
7	31:16	**LumaIntraMode[1]** Specifies the Luma Intra Prediction mode for four 4x4 sub-block of a MB, 4-bit each. See the bit assignment table later in this section.
	15:0	**LumaIntraMode[0]** Specifies the Luma Intra Prediction mode for four 4x4 sub-block, four 8x8 block or one intra16x16 of a MB. 4-bit per 4x4 sub-block (Transform8x8Flag=0, Mbtype=0 and intraMbFlag=1) or 8x8 block (Transform8x8Flag=1, Mbtype=0, MbFlag=1), since there are 9 intra modes. 4-bit for intra16x16 MB (Transform8x8Flag=0, Mbtype=1 to 24 and intraMbFlag=1), but only the LSBit[1:0] is valid, since there are only 4 intra modes. See the bit assignment table later in this section.
8	31:16	**LumaIntraMode[3]** Specifies the Luma Intra Prediction mode for four 4x4 sub-block of a MB, 4-bit each. See the bit assignment table later in this section.
	15:0	**LumaIntraMode[2]** Specifies the Luma Intra Prediction mode for four 4x4 sub-block of a MB, 4-bit each.

<table>
<tr><th>Dword</th><th>Bit</th><th>Description</th></tr>
<tr><td></td><td></td><td>See the bit assignment later in this section.</td></tr>
<tr><td>9</td><td>31:8</td><td>Reserved : MBZ

(Reserved for encocder turbo mode IntraResidueDataSize, when this is not 0, optional residue data are provided to the PAK; Reserved for decoder)</td></tr>
<tr><td></td><td>7:0</td><td>IntraStruct

This field contains 6 bits for IntraPredAvailFlags[5:0] and 2 bits for ChromaIntraPredMode. The IntraPredAvailFlags[4:0] (the lower 5 bits) have already included the effect of the constrained_intra_pred_flag. See the diagram later for the definition of neighbor position around the current MB or MB pair (in MBAFF mode).

1 – IntraPredAvailFlagY, indicates the values of samples of neighbor Y can be used in intra prediction for the current MB.

0 – IntraPredAvailFlagY, indicates the values of samples of neighbor Y is not available for intra prediction of the current MB.

IntraPredAvailFlag-A and -E can only be different from each other when constrained_intra_pred_flag is equal to 1 and mb_field_decoding_flag is equal to 1 and the value of the mb_field_decoding_flag for the macroblock pair to the left of the current macroblock is equal to 0 (which can only occur when MbaffFrameFlag is equal to 1).

IntraPredAvailFlag-F is used only if

it is in MBAFF mode, i.e. MbaffFrameFlag = 1,

the current macroblock is of frame type, i.e. MbFieldFag = 0, and

the current macroblock type is Intra8x8, i.e. IntraMbFlag = INTRA, IntraMbMode = INTRA_8x8, and Transform8x8Flag = 1.

In any other cases IntraPredAvailFlag-A shall be used instead.
<table>
<tr><th>Bits</th><th>IntraPredAvailFlags Definition</th></tr>
<tr><td>7</td><td>IntraPredAvailFlagF – F (Left 8th row (-1,7) neighbor)</td></tr>
<tr><td>6</td><td>IntraPredAvailFlagA – A (Left neighbor top half)</td></tr>
<tr><td>5</td><td>IntraPredAvailFlagE – E (Left neighbor bottom half)</td></tr>
<tr><td>4</td><td>IntraPredAvailFlagB – B (Top neighbor)</td></tr>
<tr><td>3</td><td>IntraPredAvailFlagC – C (Top right neighbor)</td></tr>
<tr><td>2</td><td>IntraPredAvailFlagD – D (Top left corner neighbor)</td></tr>
<tr><td>1:0</td><td>ChromaIntraPredMode – 2 bits to specify 1 of 4 chroma intra prediction modes, see the table in later section.</td></tr>
</table></td></tr>
</table>

Inline data subfields for an Inter Macroblock

DWord	Bit	Description
7	31:16	**Reserved : MBZ**
	15:8	**SubMbPredMode (Sub-Macroblock Prediction Mode):** If **InterMbMode** is INTER8x8, this field describes the prediction mode of the sub-partitions in the four 8x8 sub-macroblock. It contains four subfields each with 2-bits, corresponding to the four 8x8 sub-macroblocks in sequential order. This field is derived from sub_mb_type for a BP_8x8 macroblock. This field is derived from **MbType** for a non-BP_8x8 inter macroblock, and carries redundant information as **MbType**). If **InterMbMode** is INTER16x16, INTER16x8 or INTER8x16, this field carries the prediction modes of the sub macroblock (one 16x16, two 16x8 or two 8x16). The unused bits are set to zero. Bits [1:0]: SubMbPredMode[0] Bits [3:2]: SubMbPredMode[1] Bits [5:4]: SubMbPredMode[2] Bits [7:6]: SubMbPredMode[3]
	7:0	**SubMbShape** (Sub Macroblock Shape) This field describes the sub-block partitioning of each sub macroblocks (four 8x8 blocks). It contains four subfields each with 2-bits, corresponding to the 4 fixed size 8x8 sub macroblocks in sequential order. This field is provided for MB with sub_mb_type equal to BP_8x8 only (B_8x8 and P_8x8 as defined in DXVA). Otherwise, this field is ignored by hardware Bits [1:0]: SubMbShape[0] – for 8x8 Block 0 Bits [3:2]: SubMbShape[1] – for 8x8 Block 1 Bits [5:4]: SubMbShape[2] – for 8x8 Block 2 Bits [7:6]: SubMbShape[3] – for 8x8 Block 3 Blocks of the MB is numbered as follows : 01 23 Each 2-bit value [1:0] is defined as : 00 – SubMbPartWidth=8, SubMbPartHeight=8 01 – SubMbPartWidth=8, SubMbPartHeight=4 10 – SubMbPartWidth=4, SubMbPartHeight=8 11 – SubMbPartWidth=4, SubMbPartHeight=4
8	31:24	**RefPicSelect[0][3]** Support up to 4 reference pictures per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the reference index into the Reference Picture List0 Table.

DWord	Bit	Description
	23:16	**RefPicSelect[0][2]** Support up to 4 reference pictures per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the reference index into the Reference Picture List0 Table.
	15:8	**RefPicSelect[0][1]** Support up to 4 reference pictures per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the reference index into the Reference Picture List0 Table.
	7:0	**RefPicSelect[0][0]** Support up to 4 reference pictures per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the reference index into the Reference Picture List0 Table.
9	31:24	**RefPicSelect[1] [3]** Support up to 4 reference pictures per L1 direction, one per MB partition, if exists. See details in later section. This field specifies the reference index into the Reference Picture List1 Table. For P- picture these bits must be set to zero.
	23:16	**RefPicSelect[1][2]** Support up to 4 reference pictures per L1 direction, one per MB partition, if exists. See details in later section. This field specifies the reference index into the Reference Picture List1 Table. For P- picture these bits must be set to zero.
	15:8	**RefPicSelect[1][1]** Support up to 4 reference pictures per L1 direction, one per MB partition, if exists. See details in later section. This field specifies the reference index into the Reference Picture List1 Table. For P- picture these bits must be set to zero.
	7:0	**RefPicSelect[1][0]** Support up to 4 reference pictures per L1 direction, one per MB partition, if exists. See details in later section. This field specifies the reference index into the Reference Picture List1 Table. For P- picture these bits must be set to zero.

2.3.1.1.1 Luma Intra Prediction Modes

Luma Intra Prediction Modes (LumaIntraPredModes) is defined in *Luma Intra Prediction Modes*. It is further categorized as Intra16x16PredMode (*Luma Intra Prediction Modes*), Intra8x8PredMode (*Luma Intra Prediction Modes*) and Intra4x4PredMode (*Luma Intra Prediction Modes*), operating on 16x16, 8x8 and 4x4 block sizes, respectively. illustrates the intra prediction directions geometrically for the Intra4x4 prediction. When a macroblock is subdivided, the intra prediction is performed for the subdivision in a predetermined order. For example, *Luma Intra Prediction Modes* shows the block order for Intra4x4

prediction. And *Luma Intra Prediction Modes* shows the block order of Block8x8 in a 16x16 region or Block4x4 in an 8x8 region.

Definition of LumaIntraPredModes

LumaIntraPredModes [index]		Intra16x16PredMode	Intra8x8PredMode	Intra4x4PredMode
Index	Bit	MbType = [1…24] Transform8x8Flag = 0	MbType = 0 Transform8x8Flag = 1	MbType = 0 Transform8x8Flag = 0
0	15:12	MBZ	**Block8x8 3**	**Block4x4 3 (0_0)**
	11:8	MBZ	**Block8x8 2**	**Block4x4 2 (0_1)**
	7:4	MBZ	**Block8x8 1**	**Block4x4 1 (0_2)**
	3:0	**Block16x16**	**Block8x8 0**	**Block4x4 0 (0_3)**
1	15:12	MBZ	MBZ	**Block4x4 7 (1_0)**
	11:8	MBZ	MBZ	**Block4x4 6 (1_1)**
	7:4	MBZ	MBZ	**Block4x4 5 (1_2)**
	3:0	MBZ	MBZ	**Block4x4 4 (1_3)**
2	15:12	MBZ	MBZ	**Block4x4 11 (2_0)**
	11:8	MBZ	MBZ	**Block4x4 10 (2_1)**
	7:4	MBZ	MBZ	**Block4x4 9 (2 2)**
	3:0	MBZ	MBZ	**Block4x4 8 (2_3)**
3	15:12	MBZ	MBZ	**Block4x4 15 (3_0)**
	11:8	MBZ	MBZ	**Block4x4 14 (3_1)**
	7:4	MBZ	MBZ	**Block4x4 13 (3_2)**
	3:0	MBZ	MBZ	**Block4x4 12 (3_3)**

Definition of Intra16x16PredMode

Intra16x16PredMode	Description
0	**Intra_16x16_Vertical**
1	**Intra_16x16_Horizontal**
2	**Intra_16x16_DC**

Intra16x16PredMode	Description
3	**Intra_16x16_Plane**
4 – 15	Reserved

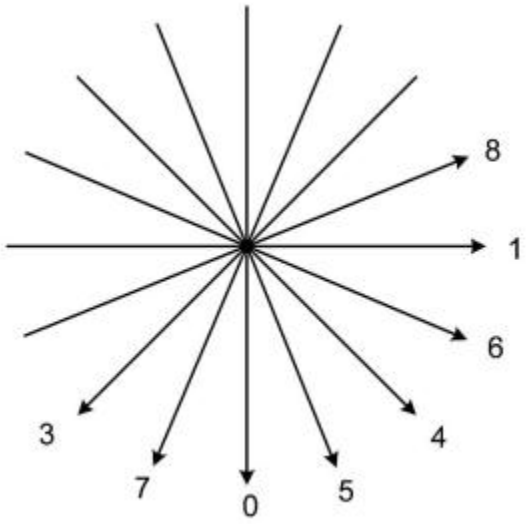
Definition of Intra8x8PredMode

Intra8x8PredMode	Description
0	**Intra_8x8_Vertical**
1	**Intra_8x8_Horizontal**
2	**Intra_8x8_DC**
3	**Intra_8x8_Diagonal_Down_Left**
4	**Intra_8x8_Diagonal_Down_Right**
5	**Intra_8x8_Vertical_Right**
6	**Intra_8x8_Horizontal_Down**
7	**Intra_8x8_Vertical_Left**
8	**Intra_8x8_Horizontal_Up**
9 – 15	Reserved

Definition of Intra4x4PredMode

Intra4x4PredMode	Description
0	**Intra_4x4_Vertical**
1	**Intra_4x4_Horizontal**
2	**Intra_4x4_DC**
3	**Intra_4x4_Diagonal_Down_Left**
4	**Intra_4x4_Diagonal_Down_Right**
5	**Intra_4x4_Vertical_Right**
6	**Intra_4x4_Horizontal_Down**
7	**Intra_4x4_Vertical_Left**
8	**Intra_4x4_Horizontal_Up**
9 – 15	Reserved

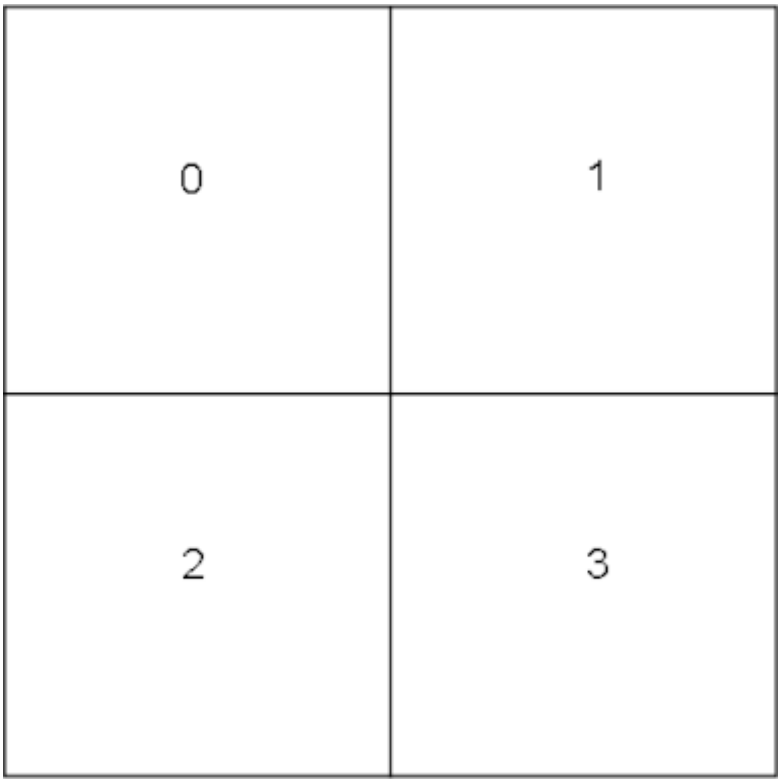
Intra_4x4 prediction mode directions



Numbers of Block4x4 in a 16x16 region

0	1	4	5
2	3	6	7
8	9	12	13
10	11	14	15

Numbers of Block4x4 in an 8x8 region or numbers of Block8x8 in a 16x16 region



Definition of Chroma Intra Prediction Mode

ChromaIntraPredMode (intra_chroma_pred_mode)	Name of intra_chroma_pred_mode
0	Intra_Chroma_DC (prediction mode)
1	Intra_Chroma_Horizontal (prediction mode)
2	Intra_Chroma_Vertical (prediction mode)
3	Intra_Chroma_Plane (prediction mode)

2.3.1.1.1 Reference Indices defined for each MB partition type and Bit Assignment

		frame/field MB/Picture			
MB partitioning	16x16	16x8	8x16	8x8	
RefIdxL0/1[0]	blk0	blk0	blk0	blk0	Bit 7:0
RefIdxL0/1[1]	x	blk1	blk1	blk1	Bit 15:8
RefIdxL0/1[2]	x	x	x	blk2	Bit 23:16
RefIdxL0/1[3]	x	x	x	blk3	Bit 31:24

2.3.1.1.2 MB Neighbor Availability in Intra-Prediction Modes (IntraPredAvailFlags)

Current MB is labelled as X. For non-MBAFF mode, 4 neighbors, A, B, C, D, are depicted in the following picture and are defined as the following.

- MB D: top left neighbor of current MB X
- MB C: top right neighbor of current MB X
- MB B: top neighbor of current MB X
- MB A: left neighbor of the current MB X

mbAddrDD (top-left)	**mbAddrBB** (top)	**mbAddrCC** (top-right)
mbAddrAA (left)	**X** **CurrMbAddrX**	**N/A**
N/A	**N/A**	**N/A**

For MBAFF mode, the current MB is labelled as X0 or X1, 4 neighbor pairs, A0/A1, B0/B1, C0/C1, D0/D1, are depicted in the following picture and are defined as the following.

- MB D0: first MB of top left neighbor MB pair of current MB pair X0/X1
- MB D1: second MB of top left neighbor MB pair of current MB pair X0/X1
- MB C0: first MB of top right neighbor MB pair of current MB pair X0/X1
- MB C1: second MB of top right neighbor MB pair of current MB pair X0/X1
- MB B0: first MB of top neighbor MB pair of current MB pari X0/X1
- MB B1: second MB of top neighbor MB pair of current MB pari X0/X1
- MB A0: first MB of left neighbor MB pair of the current MB pair X0/X1
- MB A1: second MB of left neighbor MB pair of the current MB pair X0/X1

mbAddrDD0	mbAddrBB0	mbAddrCC0
mbAddrD+1 D1	mbAddrB+1 B1	mbAddrC+1 C1
mbAddrAA0	CurrMbAddrX X0 or	N/A
mbAddrA+1 A1	CurrMbAddrX X1	N/A

For a given macroblock X (or X0/X1), the 6 neighbor availability signals, namely, A, B, C, D, E, F, are defined as the following.

- IntraPredAvailFlagF – F: (Single neighbor pixel at the left 8th row (-1,7)
- IntraPredAvailFlagA – A (Left neighbor top half pixel group)
- IntraPredAvailFlagE – E (Left neighbor bottom half pixel group)
- IntraPredAvailFlagB – B (Top neighbor pixel group)
- IntraPredAvailFlagC – C (Top right neighbor pixel group)
- IntraPredAvailFlagD – D (Top left corner neighbor pixel)

The following table depicts the generation of IntraPredAvailFlags[5:0] signals in a condensed form. It should note that for most cases only one input neighbor signal is assigned for each condition. The exception is in the four places for deriving left neighbor A and E where the neighbor is only available if left neighbors (A0 and A1) are both available (A0&A1). Also note that F takes output value very similar to that for A except the two "AND" conditions, where F is assigned to A1 instead of (A0&A1).

Definition of intra-prediction neighbor availability calculation in MBAFF mode

Output ➔		D		B		C		A		E		F	
Current X \ Neighbor Y		Y-Frame	Y-Field	Y-Frame	Y-Field	Y-Frame	Y-Field	Y-Frame	Y-Field	Y-Frame	Y-Field	Y-Frame	Y-Field
X_0 (Top)	X-Frame	**D_1**	**D_1**	**B_1**	**B_1**	**C_1**	**C_1**	**A_0**	**A_0 & A_1**	**A_0**	**A_0 & A_1**	**A_0**	**A_1**
	X-Field	**D_1**	**D_0**	**B_1**	**B_0**	**C_1**	**C_0**	**A_0**	**A_0**	**A_1**	**A_0**	**A_0**	**A_0**
X_1 (Bottom)	X-Frame	**A_0**	**A_1**	**X_0**	N/A	0	0	**A_1**	**A_0 & A_1**	**A_1**	**A_0 & A_1**	**A_1**	**A_1**
	X-Field	**D_1**	**D_1**	**B_1**	**B_1**	**C_1**	**C_1**	**A_0**	**A_1**	**A_1**	**A_1**	**A_0**	**A_1**

In *MB Neighbor Availability in Intra Prediction Modes IntraPredAvailFlags*, *X-Frame* or *X-Field* indicates the frame/field mode of the current MB; and *Y-Frame* or *Y-Field* indicates the corresponding neighbor MB for the given neighbor location, being upper left (D) or left (A) for example. Therefore, "Y-" takes the selected neighbor MB name as in the output cell in the same column. For example, for output D, if X1 is a frame MB, Y = A, if X1 is a field MB, Y = D.

For non-MBAFF mode, as A0=A1, B0=B1, C0=C1 and D0=D1, the neighbor assignment is degenerated into the following simple table. Here, E is assigned to the same as A and F is forced to 0.

Definition of intra-prediction neighbor availability calculation in non-MBAFF mode

Output ➔	D	B	C	A	E	F
X	D0	B0	C0	A0	A0	0

To further explain the neighbor assignment rules in *MB Neighbor Availability in Intra Prediction Modes IntraPredAvailFlags*, the following table provides description for each condition. Please note that this table is **informative** as it provides redundant information as in *MB Neighbor Availability in Intra Prediction Modes IntraPredAvailFlags*.

Detailed explanation of intra-prediction neighbor availability calculation in MBAFF mode

Current MB	Current MB Field	Neighbor MB Field	Neighbor MB Select (Y=?)	Neighbor Avail Result (OUTPUT)	Description
				D	
X0	X-Frame	Y-Frame	D	D1	Top Frame MB uses [-1,-1] = D_31, thus D1 only, regardless D frame or field pair
(Top)	X-Frame	Y-Field	D	D1	
	X-Field	Y-Frame	D	D1	Top Field MB uses [-1,-2] = D_30, thus if D is frame pair, takes D1 (D1_14 pixel), and if D is field pair, takes D0 (D0_15 pixel)
	X-Field	Y-Field	D	D0	
X1	X-Frame	Y-Frame	A	A0	Bottom Frame MB uses [-1,15] = A_15, thus A0 (A0_15 pixel) if A is a frame pair, or A1 (A1_7 pixel), if A is a field pair
(Bottom)	X-Frame	Y-Field	A	A1	
	X-Field	Y-Frame	D	D1	Bottom Field MB uses [-1,-1] = D_31, thus D1 only, regardless D frame or field pair
	X-Field	Y-Field	D	D1	
				B	
X0	X-Frame	Y-Frame	B	B1	Top Frame MB uses [0…15,-1] = B_31, thus B1 only, regardless B frame or field pair
(Top)	X-Frame	Y-Field	B	B1	
	X-Field	Y-Frame	B	B1	Top Field MB uses [0…15,-2] = B_30, thus if B is frame pair, takes B1 (B1_14 row), and if B is field pair, takes B0 (B0_15 row)
	X-Field	Y-Field	B	B0	
X1	X-Frame	Y-Frame	X	X0	Bottom Frame MB uses [0…15,15], thus X0 (X0_15 row)
(Bottom)	X-Frame	Y-Field	X	n/a	*Note: X0 and X1 must have the same field type, this row is n/a.*
	X-Field	Y-Frame	B	B1	Bottom Field MB uses [0…15,-1] = B_31, thus B1 only, regardless B frame or field pair
	X-Field	Y-Field	B	B1	
				C	
X0	X-Frame	Y-Frame	C	C1	Top Frame MB uses [16…23,-1] = C_31, thus C1 only, regardless C frame or field pair
(Top)	X-Frame	Y-Field	C	C1	
	X-Field	Y-Frame	C	C1	Top Field MB uses [16…23,-2] = C_30, thus if C is frame pair, takes C1 (C1_14 row), and if C is field pair, takes C0 (C0_15 row)
	X-Field	Y-Field	C	C0	
X1	X-Frame	Y-Frame	n/a	0	Bottom Frame MB doesn't have left-top neighbor by definition, thus forced to 0
(Bottom)	X-Frame	Y-Field	n/a	0	
	X-Field	Y-Frame	C	C1	Bottom Field MB uses [16…23,-1] = C_31, thus C1 only, regardless C frame or field pair
	X-Field	Y-Field	C	C1	
				A	
X0	X-Frame	Y-Frame	A	A0	First Half of Top Frame MB uses [-1,0…7], thus A0 if A is a frame pair; but is only avail if both A0 and A1 are avail if A is a field pair due to the mix
(Top)	X-Frame	Y-Field	A	A0&A1	
	X-Field	Y-Frame	A	A0	First Half of Top Field MB uses [-1,0..2..4..14], thus take A0 (if A is frame pair, takes A0 even lines, and if A is field pair, takes A0 first half)
	X-Field	Y-Field	A	A0	
X1	X-Frame	Y-Frame	A	A1	First Half of Bottom Frame MB uses [-1,16…23], thus A1 if A is a frame pair; but is only avail if both
(Bottom)	X-Frame	Y-Field	A	A0&A1	

Current MB	Current MB Field	Neighbor MB Field	Neighbor MB Select (Y=?)	Neighbor Avail Result (OUTPUT)	Description
				D	
					A0 and A1 are avail if A is a field pair due to the mix
	X-Field	Y-Frame	A	A0	First Half of Bottom Field MB uses [-1,1..3..15], thus take A0 (if A is frame pair, takes A0 odd lines, and if A is field pair, takes A1 first half)
	X-Field	Y-Field	A	A1	
				E	
X0	X-Frame	Y-Frame	A	A0	Second Half of Top Frame MB uses [-1,8…15], thus A0 if A is a frame pair; but is only avail if both A0 and A1 are avail if A is a field pair due to the mix
(Top)	X-Frame	Y-Field	A	A0&A1	
	X-Field	Y-Frame	A	A1	Second Half of Top Field MB uses [-1,16..18..30], thus take A1 (if A is frame pair, takes A1 even lines, and if A is field pair, takes A0 second half)
	X-Field	Y-Field	A	A0	
X1	X-Frame	Y-Frame	A	A1	Second Half of Bottom Frame MB uses [-1,24…31], thus A1 if A is a frame pair; but is only avail if both A0 and A1 are avail if A is a field pair due to the mix
(Bottom)	X-Frame	Y-Field	A	A0&A1	
	X-Field	Y-Frame	A	A1	Second Half of Bottom Field MB uses [-1,17..19..31], thus takes A1 (if A is frame pair, takes A1 odd lines, and if A is field pair, takes A1 second half)
	X-Field	Y-Field	A	A1	
				F	
X0	X-Frame	Y-Frame	A	A0	Top Frame MB uses [-1,7] = A_7 (odd location), thus A0 if A is frame pair and A1 if field pair
(Top)	X-Frame	Y-Field	A	A1	
	X-Field	Y-Frame	A	A0	Top Field MB uses [-1,14] = A_14 (even location), thus A0 regardless A frame or field pair
	X-Field	Y-Field	A	A0	
X1	X-Frame	Y-Frame	A	A1	Bottom Frame MB uses [-1,23] = A_23 (odd location), thus A1 regardless A frame or field pair
(Bottom)	X-Frame	Y-Field	A	A1	
	X-Field	Y-Frame	A	A0	Bottom Field MB uses [-1,15] = A_15 (odd location), thus A0 if A is frame pair and A1 if A is field pair
	X-Field	Y-Field	A	A1	

2.3.1.1.3 Macroblock Type for Intra Cases

MbType follows two different tables according to whether the macroblock is an inter or intra macroblock according to IntraMbFlag.

For an intra macroblock, MbType, as defined in *Macroblock Type for Intra Cases*, carries redundant information as IntraMbMode. The notation I_16x16_x_y_z used in the table, ‘x’ is Intra16x16LumaPredMode, ‘y’ is ChromaCbpInd, and ‘z’ is LumaCbpInd, as defined in *Macroblock Type for Intra Cases*.

MbType definition for Intra Macroblock

Macroblock Type	MbType
I_4x4	0
I_8x8	0

Macroblock Type	MbType
I_16x16_0_0_0	1
I_16x16_1_0_0	2
I_16x16_2_0_0	3
I_16x16_3_0_0	4
I_16x16_0_1_0	5
I_16x16_1_1_0	6
I_16x16_2_1_0	7
I_16x16_3_1_0	8
I_16x16_0_2_0	9
I_16x16_1_2_0	Ah
I_16x16_2_2_0	Bh
I_16x16_3_2_0	Ch
I_16x16_0_0_1	Dh
I_16x16_1_0_1	Eh
I_16x16_2_0_1	Fh
I_16x16_3_0_1	10h
I_16x16_0_1_1	11h
I_16x16_1_1_1	12h
I_16x16_2_1_1	13h
I_16x16_3_1_1	14h
I_16x16_0_2_1	15h
I_16x16_1_2_1	16h
I_16x16_2_2_1	17h
I_16x16_3_2_1	18h
I_PCM	19h (used by HW)

Note: MbType here is identical as specified in DXVA 2.0.

For Intra_16x16 modes, the 5 bits of value (MbType – 1) have the following meanings.

Sub field definition used by MbType for a macroblock with Intra16x16 prediction

Bits	Description
4	**LumaCbpInd** – Luma Coded Block Pattern Indicator 0 means none of the luma blocks are coded. 1 means that at least one luma block is coded. 0 = SUBMODE_I16_L_0 1 = SUBMODE_I16_L_NZ In VME output, this field is forced to be 1 before adding 1 in Intra_16x16 mode.
3:2	**ChromaCbpInd** – Chroma Coded Block Pattern Indicator 00 means none of chroma blocks are coded. 01 means that only the chroma DC block is coded, but all AC blocks are not coded. 10 means that at least one AC chroma block is coded. 00 = SUBMODE_I16_C_0 01 = SUBMODE_I16_C_DC 10 = SUBMODE_I16_C_NZ 11 = Reserved In VME output, this field is forced to be 10 before adding 1 in Intra_16x16 mode. *Programming Note: Adding 1 to MbType by VME hardware may have carry in to this field. But as '11' is reserved, the carry-in doesn't propagate into bit 4 or higher. This allows software to update MbType, if desired, using the redundant LumaIntraPredModes information.*
1:0	**Intra16x16PredMode** – Intra16x16 Prediction Mode These two bits carries redundant (identical) information as that in LumaIntraPredModes[0][0]. 0 = SUBMODE_I16_VER 1 = SUBMODE_I16_HOR 2 = SUBMODE_I16_DC 3 = SUBMODE_I16_PLANE

IntraMbMode definition

IntraMbMode [1:0]	Description	Supported by VME?	Used by PAK?
0	**INTRA_16x16 (redundant with MbType)**	Yes	Ignored
1	**INTRA_8x8**	Yes	Yes
2	**INTRA_4x4**	Yes	Yes
3	**IPCM (redundant with MbType)**	No	Ignored

As an alternative representation, MbType is logically the same as the following, except the I_PCM and I_NxN (i.e. I_4x4 and I_8x8) cases:

- 24 types of 16x16 intra modes: **A**+**B**+**C**+**D**: (1h – 18h)

MBTYPE_INTRA_16x16	1h	A

- 4 Intra16x16 modes:

SUBMODE_I16_VER	0	B
SUBMODE_I16_HOR	1	B
SUBMODE_I16_DC	2	B
SUBMODE_I16_PLN	3	B

- 3 Chroma Cbp indices:

SUBMODE_I16_C_0	0	C
SUBMODE_I16_C_DC	4	C
SUBMODE_I16_C_NZ	8	C

- 2 Luma Cbp indices:

SUBMODE_I16_L_0	0	D
SUBMODE_I16_L_NZ	Ch	D

2.3.1.1.4 Macroblock Type for Inter Cases

Sub-Macroblock Prediction Mode, SubMbPredMode, indicates the prediction mode for the sub-partitions. Prediction mode specifies prediction direction being forward (from L0), backward (from L1) or bi-directional (from both L0 and L1). Its meaning depends on InterMbMode. *Macroblock Type for Inter Cases* provides the definition of the field.

- If InterMbMode is INTER16x16, only SubMbPredMode[0] is valid, it describes the prediction mode of the 16x16 macroblock. The other entries are set to zero by hardware.
  - For AVC, SubMbPredMode[0] contains redundant information as encoded in MbType parameter.
  - *Note: SubMbPredMode[1]-[3] are intentionally set to zero to allow a simple LUT to derive MbType as described later.*
- If InterMbMode is INTER16x8, and INTER8x16, only the first two entries SubMbPredMode[0] and SubMbPredMode[1] are valid, describing the sub-macroblock prediction mode.
  - For AVC, SubMbPredMode[0]/[1] contains redundant information as encoded in MbType parameter.
  - *Note: SubMbPredMode[2]-[3] are intentionally set to zero to allow a simple LUT to derive MbType as described later.*
- If InterMbMode is INTER8x8, each entry of SubMbPredMode describes the prediction mode of the sub-partition of an 8x8 sub-macroblock.
  - For AVC, SubMbPredMode can be derived from sub_mb_type field for BP_8x8 macroblocks as defined in AVC spec.
  - *Note on Direct Sub-macroblock Prediction Mode: Direct prediction is not conveyed through SubMbPredMode, instead, it is carried through Direct8x8Pattern.*

InterMbMode definition

MbSkipFlag	InterMbMode	Description
0	0	**INTER16x16**

MbSkipFlag	InterMbMode	Description
0	1	**INTER16x8**
0	2	**INTER8x16**
0	3	**INTER8x8**
1	0	**PSKIP/BSKIP16x16***
1	3	**BSKIP**
1	1, 2	Reserved
Used by PAK	Ignored by PAK	

* BSKIP16x16 is an optional non-standard but equivalent optimization.

Definition of SubMbPredMode based on InterMbMode

SubMbPredMode	INTER16x16	INTER16x8	INTER8x16	INTER8x8
Bit	MbType = [1…3]	MbType = [16h]	MbType = [4…15h]	MbType = [16h]
7:6	MBZ	MBZ	MBZ	**Block8x8 3**
5:4	MBZ	MBZ	MBZ	**Block8x8 2**
3:2	MBZ	**Block16x8 1**	**Block8x16 1**	**Block8x8 1**
1:0	**Block16x16**	**Block16x8 0**	**Block8x16 0**	**Block8x8 0**
	Ignored by PAK	**Ignored by PAK**	**Ignored by PAK**	**Used by PAK**

Definition of SubMbPredMode[i]

SubMbPredMode	Description	InterMbMode	VME Output	MvCountPred	Notes
0	Pred_L0	All	Yes	1	P or B Slice
1	Pred_L1	All	Yes	1	B Slice Only
2	BiPred	All	Yes	2	B Slice Only
3	Reserved	Reserved	Reserved	Reserved	Reserved

Sub-Macroblock Shape, SubMbShape[i], for i = 0…3, describes the shape of the sub partitions of the 8x8 sub-macroblock of a BP_8x8 macroblock. This field is only valid if InterMBMode is INTER8x8. They are defined in *Macroblock Type for Inter Cases*. The parameters can be derived from *sub_mb_type* field as defined in AVC spec.

Note: These fields must be correctly set even for **Direct** or **Skip** 8x8 cases, the individual B_Direct_8x8 block is flagged by the **Direct8x8Pattern** variable.

Definition of SubMbShape for an 8x8 region of a BP_8x8 macroblock (including BSKIP, BDIRECT)

	Description			
SubMbShape	NumSubMbPart	SubMbPartWidth	SubMbPartHeight	MvCountShape
0	1	8	8	1
1	2	8	4	2
2	2	4	8	2
3	4	4	4	4

For an inter macroblock, MbType, carries redundant information as InterMbMode and SubMbPredMode. *Macroblock Type for Inter Cases* provides the typical inter macroblock types and *Macroblock Type for Inter Cases* provides that with skip and direct modes. The definition of MbType for both P slice and B slice is the same and is equivalent to that for mb_type of a B slice in the AVC spec. As direct mode is indicated using a separate field Direct8x8Pattern, 0 is reserved for MbType.

Here, MVCount is the number of motion vectors actually encoded in the bitstream. It is informative. For a BP_8x8 or equivalent Skip/Direct macroblock, MVCount is the sum of the following term for the four 8x8 sub macroblock (with i = 0…3):

MvCountShape[i] * MvCountPred[i] * MvCountDirect[i]

where MvCountShape[i] is block count for sub macroblock [i], MvCountPred[i] is the motion vector count for each block of sub macroblock[i], and MvCountDirect[i] is the multipler for direct mode for B Slice, indicating whether motion vectors are coded or not. It must be set to 1 for P slice. For B Slice, MvCountDirect[i] = !Direct8x8Pattern[i], which is 0 for a sub macroblock coded as direct mode and 1 otherwise.

In the tables, “DC” stands for “Don’t Care” as PAK hardware ignores these fields.

MbType definition for Inter Macroblock (and MbSkipflag = 0)

Macroblock Type	MbType	MbSkipFlag	Direct8x8Pattern	SubMbShape	SubMbPredMode	MVCount
Reserved	0	-	-	-	-	-
BP_L0_16x16	1	0	0	DC	DC	1
B_L1_16x16	2	0	0	DC	DC	1
B_Bi_16x16	3	0	0	DC	DC	2
BP_L0_L0_16x8	4	0	0	DC	DC	2
BP_L0_L0_8x16	5	0	0	DC	DC	2
B_L1_L1_16x8	6	0	0	DC	DC	2
B_L1_L1_8x16	7	0	0	DC	DC	2
B_L0_L1_16x8	8	0	0	DC	DC	2
B_L0_L1_8x16	9	0	0	DC	DC	2

Macroblock Type	MbType	MbSkipFlag	Direct8x8Pattern	SubMbShape	SubMbPredMode	MVCount
B_L1_L0_16x8	0Ah	0	0	DC	DC	2
B_L1_L0_8x16	0Bh	0	0	DC	DC	2
B_L0_Bi_16x8	0Ch	0	0	DC	DC	3
B_L0_Bi_8x16	0Dh	0	0	DC	DC	3
B_L1_Bi_16x8	0Eh	0	0	DC	DC	3
B_L1_Bi_8x16	0Fh	0	0	DC	DC	3
B_Bi_L0_16x8	10h	0	0	DC	DC	3
B_Bi_L0_8x16	11h	0	0	DC	DC	3
B_Bi_L1_16x8	12h	0	0	DC	DC	3
B_Bi_L1_8x16	13h	0	0	DC	DC	3
B_Bi_Bi_16x8	14h	0	0	DC	DC	4
B_Bi_Bi_8x16	15h	0	0	DC	DC	4
BP_8x8	16h	0	!= Fh	vary	vary	Sum
Reserved	17h-1Fh	-	-	-	-	-

Additional MbType definition with Direct/Skip for Inter Macroblock

Macroblock Type	Mb Type	Xfrm 8x8	MbSkip Flag	Direct8x8 Pattern	SubMb Shape	SubMb PredMode	MvCount	Notes
P_Skip_16 x16	1	-	1	DC	DC	DC	0	Skipped macroblock. Motion compensation like P_L0_16x16
B_Skip_16 x16_4MVP air	**16h**	**Vary**	**1**	**Fh**	**0**	**vary**	**0**	Skipped macroblock. Motion compensation like B_8x8 with 8x8 subblocks, when **direct_8x8_inference_flag** is set to 1
B_Skip_16 x16_16MV Pair	16h	0	1	Fh	FFh	vary	0	Skipped macroblock. Motion compensation like B_8x8 with 4x4 subblocks, when **direct_8x8_inference_flag** is set to 0
B_Direct_1	16h	vary	0	Fh	0	vary	0	MbType coded as

Macroblock Type	Mb Type	Xfrm 8x8	MbSkip Flag	Direct8x8 Pattern	SubMb Shape	SubMb PredMode	MvCount	Notes
6x16_4MV Pair								B_Direct_16x16. Motion compensation like B_8x8 with 8x8 subblocks, when **direct_8x8_inference_flag** is set to 1
B_Direct_16x16_16MV Pair	16h	0	0	Fh	FFh	vary	0	MbType coded as B_Direct_16x16. Motion compensation like B_8x8 with 4x4 subblocks, when **direct_8x8_inference_flag** is set to 0

People might notice that B_DIRECT_16x16 and B_SKIP are mapped on BP_8x8 for AVC decoding interface in IT mode as the motion compensation operation for both modes are the same as BP_8x8. According to AVC Spec, motion vectors for B_DIRECT_16x16 and B_SKIP are derived from temporally co-located macroblock on an 8x8 sub macroblock basis if direct_8x8_inference_flag is set to 1 or on a 4x4 block basis if it is set to 0. For each sub macroblock or block, SubMbPredMode is derived, thus can any of the valid numbers. Motion vectors may also be different. In spatial direct mode, the motion vectors are subject to spatial neighbor macroblocks as well as co-located macroblock. The spatial prediction is based on the neighbor macroblocks, so the same spatial predicted motion vector applies to all sub macroblocks or blocks. However, under certain conditions, temporal predictor may replace (colZeroFlag) the spatial predictor for a given sub macroblock or block. Thus the motion vectors may differ.

In *Macroblock Type for Inter Cases*, the macroblock type names for major partitions nicely follow forms *BP_MbPredMode_MbShape* (like BP_L0_16x16) and *B_MbPredMode0_MbPredMode1_MbShape* (like B_L0_Bi_16x8). For minor partitions it is fixed as *BP_MbShape* as BP_8x8.

However, in *Macroblock Type for Inter Cases* the macroblock types for Skip and Direct modes does not follow the same rule. The third field in P_Skip_16x16 or B_Direct_16x16_x indicates that “Skip” or “Direct” applies to the entire 16x16 macroblock, even though MbShape is 8x8 as that in BP_8x8. In order to distinguish the SubMbShape being 8x8 or 4x4 for B_Skip and B_Direct, the fourth field is added. 4MVP**air** indicates upto 4 MV pairs are presented with SubMbShape equals to 0; and 16MVP**air** indicates up to 16 MV pairs are presented with SubMbShape equals to FFh.Also note that P_8x8ref0 is not specified in PAK input interface, it is up to hardware to detect and choose its packing format based on number of reference indices and reference index for the given macroblock.

2.3.1.1.5 Macroblock Type Conversion Rules

For improved coding efficiency the PAK hardware has the capability to convert macroblock types to use more efficiency coding modes such as DIRECT and SKIP. For an inter macroblock or a sub macroblock coded as DIRECT, no motion vector is needed in the bitstream for the macroblock or sub macroblock. If a macroblock is coded as SKIP, it only consumes one SKIP bit (no motion vector, no coefficients are coded). And infomaton about the macroblock is ‘inferred’ according to the rules stated in the AVC Spec.

As the input to PAK, the following signals can convey the information regarding DIRECT and SKIP:

- MbSkipFlag
- Direct8x8Pattern
- CodecBlockPattern (CbpY, CbpCb, CbpCr)

Such conversion can be enabled or disabled through the SLICE_STATE fields **DirectConvDisable** and **SkipConvDisable** as well as the in line command field **MbSkipConvDisable**.

A P slice doesn't support direct mode, it only supports P_Skip, which is equivalent to a 16_16_L0 prediction. Other prediction types cannot be converted to P_Skip. The following table describes the macroblock type conversion rules for a P slice. Here CBP = CbpY/CbpCb/CbpCr are the final computed results after quantization by the hardware. Note that hardware honors the input CbpY/CbpCb/CbpCr fields – if the value corresponding to a block is set to zero, the resulting CBP is also zero. The output **mb_skip_flag** and **mb_type** are the symbols coded in the bitstream as defined in the AVC spec. "DC" stands for "Don't care", "T" for "True".

Note that the internal condition of MV==MVP is subject to the precise rules stated in the AVC Spec as quoted below. Note that there are exceptions for P_Skip from the normal motion vector prediction rules.

Derivation process for luma motion vectors for skipped macroblocks in P and SP slices

This process is invoked when mb_type is equal to P_Skip.

Outputs of this process are the motion vector mvL0 and the reference index refIdxL0.

The reference index refIdxL0 for a skipped macroblock is derived as follows.

refIdxL0 = 0. (8-168)

For the derivation of the motion vector mvL0 of a P_Skip macroblock type, the following applies.

– The process specified in subclause 8.4.1.3.2 is invoked with mbPartIdx set equal to 0, subMbPartIdx set equal to 0, currSubMbType set equal to "na", and listSuffixFlag set equal to 0 as input and the output is assigned to mbAddrA, mbAddrB, mvL0A, mvL0B, refIdxL0A, and refIdxL0B.

– The variable mvL0 is specified as follows.

– If any of the following conditions are true, both components of the motion vector mvL0 are set equal to 0.

– mbAddrA is not available

– mbAddrB is not available

– refIdxL0A is equal to 0 and both components of mvL0A are equal to 0

– refIdxL0B is equal to 0 and both components of mvL0B are equal to 0

– Otherwise, the derivation process for luma motion vector prediction as specified in subclause 8.4.1.3 is invoked with mbPartIdx = 0, subMbPartIdx = 0, refIdxL0, and currSubMbType = "na" as inputs and the output is assigned to mvL0.

NOTE – The output is directly assigned to mvL0, since the predictor is equal to the actual motion vector.

Macroblock type conversion rule for an inter macroblock in a P slice

Input		Internal			Output		Notes
Macroblock Type	SkipConvDisable || SkipConvDisable	CBP	MV == MVP	MbAffSkipAllowed	mb_skip_flag	mb_type	
P_Skip_16x16	DC	DC	DC	1	1	-	**Forced to P_Skip**; Hardware will force CBP to zero and also ignore SkipConvDisable control. Hardware doesn't check for MV==MVP error condition

Input		Internal			Output		Notes
Macroblock Type	SkipConvDisable || SkipConvDisable	CBP	MV == MVP	MbAffSkipAllowed	mb_skip_flag	mb_type	
P_Skip_16x16	DC	DC	DC	0	0	0	**Reverse convert to P_L0_16x16**; Hardware will force CBP to zero but reversely convert MbType as P_L0_16x16 once it determines that Skip is not allowed.
BP_16x16_L0	0	0	T	1	1	-	**Converted to P_Skip**. Even input doesn't provide skip hint, hardware can performance the optimization by detecting CBP and MV==MVP condition.
BP_16x16_L0	0	0	T	0	0	0	**Reverse back to P_L0_16x16**; Hardware will reverse back to P_L0_16x16 even Skip conditions are met once it determines that Skip is not allowed.
BP_16x16_L0	1	0	T	T	0	0	Still coded as P_L0_16x16 = 0.

A B slice supports both direct and skip modes. The following table describes the macroblock type conversion rules for a B slice. Hardware does not verify MV==MVP condition for a Skip/Direct macroblock in a B Slice as no DMV is performed by hardware.

Macroblock type conversion rule for an inter macroblock in a B slice

Input			Internal			Output		Notes
Macroblock Type	SkipConvDisable || SkipConvDisable	DirectConvDisable	CBP	MV == MVP	MbAffSkipAllowed	mb_skip_flag	mb_type	
B_Skip_8x8 B_Skip_4x4	DC	DC	DC	n/a	1	1	-	**Forced to B_Skip**; Hardware will force CBP to zero and also ignore SkipConvDisable control.
B_Skip_8x8 B_Skip_4x4	DC	DC	DC	n/a	0	0	0	**REVERSE convert to B_Direct_16x16**; Hardware will force CBP to zero and also reverse convert to B_Direct_16x16 when it

Input			Internal			Output		Notes
Macroblock Type	SkipConvDisable || SkipConvDisable	DirectConv Disable	CBP	MV == MVP	MbAffSkip Allowed	mb_skip_flag	mb_type	
								discovers Skip is not allowed.
B_Direct_16x16_4MVPair/16MVPair	0	0	0	n/a	1	1	-	**Converted to B_Skip**. Hardware first converts to B_Direct_16x16 and then further to B_Skip if CBP = 0.
B_Direct_16x16_4MVPair/16MVPair	0	0	0	n/a	0	0	0	**Converted to B_Direct_16x16**. Hardware first converts to B_Direct_16x16 and stop there as it discovers Skip is not allowed even CBP=0.
B_Direct_16x16_4MVPair/16MVPair	1	0	0	n/a	DC	0	0	**Converted to B_Direct_16x16**. Hardware converts to B_Direct_16x16 and stops there even though CBP = 0 as input disallows Skip conversion.
B_Direct_16x16_4MVPair/16MVPair	DC	0	NZ	n/a	DC	0	0	**Converted to B_Direct_16x16**. Hardware converts to B_Direct_16x16 and stops there because CBP != 0.
B_Direct_16x16_4MVPair/16MVPair	DC	1	DC	n/a	DC	0	16h	

The internal signal **MbAffSkipAllowed** is added to deal with a restriction on the frame/field flag (**MbFieldFlag**) which is unique to MBAFF. **MbAffSkipAllowed** is always set to 1 in non-MBAFF modes. In MBAFF mode, a macroblock pair may be both skipped only if its **MbFieldFlag** is the same as its

available neighbor macroblock pair A or B if A or B is available (in that order), or is not 0 if A/B are both not available. Otherwise, one of the macroblocks in the pair must be coded.

To reduce the burden on software, PAK hardware handles the above restriction correctly. For the first MB in a pair, **MbAffSkipAllowed** is always set to 1. Therefore, hardware allows converting the first MB to Skip if skip conversion is enabled. For the second MB in a pair, hardware sets **MbAffSkipAllowed** to 0 if the following is true:

- The current MB Pair has different **MbFieldFlag** than its available neighbor A or B if A or B is available, or is not 0 if A/B are both not available
- And the first MB is coded as a SKIP (could be forced or converted)

Otherwise, it sets **MbAffSkipAllowed** to 1. As **MbAffSkipAllowed** is to 0 for the above condition, hardware will disallow Skip mode for the second MB. If the input signal forces it to Skip, hardware performs reverse-convertion to code it as P_L0_16x16 or B_Direct_16x16 with CBP = 0 for a macroblock in a P or B Slice. This means that hardware is able to correct the programming mistake by software. If the macroblock is not forced to skip, hardware simply disallows Skip conversion.

Software still has an option to disallow Skip Conversion on a per-MB basis using the **MbSkipConvDisable** control field in the inline command.

2.3.1.2 Indirect Data Description

For each macroblock, an ENC-PAK data set consists of two types of data blocks: indirect **MV data block** and **inline MB information**.

The indirect MV data block may be in two modes: **unpackedmode** and **packed-size mode**.

2.3.1.2.1 Unpacked Motion Vector Data Block

In the **unpacked** mode, motion vectors are expanded (or duplicated) to either bidirectional 8x8 8MV major partition format, or bidirectional 4x4 32MV format. Thus either 32 bytes or 128 bytes is assigned to each MB.

Motion Vector block contains motion vectors in an intermediate format that is partially expanded according to the sub- macroblock size. During the expansion, a place that does not contain a motion vector is filled by replicating the relevant motion vector according to the following motion vector replication rules. If the relevant motion vector doesn't exist (for the given L0 or L1), it is zero filled.

Motion Vector Replication Rules:

- Rule #1
    - #1.1: For L0 MV, for any sub-macroblock or sub-partition where there is at least one motion vector
        - If L0 inter prediction exists, the corresponding L0 MV is used
        - Else it must be zero
    - #1.2: For L1 MV, for any sub-macroblock or sub-partition where there is at least one motion vector
        - If L1 inter prediction exists, the corresponding L1 MV is used
        - Else it must be zero
- For a macroblock with a 16x16, 16x8 or 8x16 sub-macroblock, MvSize = 8. The eight MV fields follow Rule #1.

- The 16x16 is broken down into 4 8x8 sub-macroblocks. The 16x16 MVs (after rule #1) are replicated into all 8x8 blocks.
- For an 8x16 partition, each 8x16 is broken down into 2 8x8 stacking vertically. The 8x16 MVs (after rule #1) are replicated into both 8x8 blocks.
- For a 16x8 partition, each 16x8 is broken down into 2 8x8 stacking horizontally. The 16x8 MVs (after rule #1) are replicated into both 8x8 blocks.

- For macroblock with sub-macroblock of 8x8 without minor partition (SubMbShape[0…3] = 0), MvSize = 8, (e.g. mb_type equal to P_8x8, P_8x8ref0, or B_8x8)
  - There is no motion vector replication
- For macroblock with sub-macroblock of 8x8 with at least one minor partition (if any SubMbShape[i] != 0), MvSize = 32, (e.g. mb_type equal to P_8x8, P_8x8ref0, or B_8x8)
  - For an 8x8 sub-partition, the 8x8 MVs (after rule #1) is replicated into all the four 4x4 blocks.
  - For an 4x8 sub-partition within an 8x8 partition, each 4x8 is broken down into 2 4x4 stacking vertically. The 4x8 MVs (after rule #1) are replicated into both 4x4 blocks.
  - For an 8x4 sub-partition within an 8x8 partition, each 8x4 is broken down into 2 4x4 stacking horizontally. The 8x4 MVs (after rule #1) are replicated into both 4x4 blocks.
  - For a 4x4 sub-partition within an 8x8 partition, each 4x4 has its own MVs (after rule #1).

Motion Vector block and MvSize

	DWord	Bit	MvSize	
			8	32
W1.0		31:16	**MV_Y0_L0.y**	**MV_Y0_0_L0.y**
		15:0	**MV_Y0_L0.x**	**MV_Y0_0_L0.x**
W1.1		31:16	**MV_Y0_L1.y**	**MV_Y0_0_L1.y**
		15:0	**MV_Y0_L1.x**	**MV_Y0_0_L1.x**
W1.2		31:0	**MV_Y1_L0**	**MV_Y0_1_L0**
W1.3		31:0	**MV_Y1_L1**	**MV_Y0_1_L1**
W1.4		31:0	**MV_Y2_L0**	**MV_Y0_2_L1**
W1.5		31:0	**MV_Y2_L1**	**MV_Y0_2_L0**
W1.6		31:0	**MV_Y3_L0**	**MV_Y0_3_L0**
W1.7		31:0	**MV_Y3_L1**	**MV_Y0_3_L1**
W2.0		31:0	**n/a**	**MV_Y1_0_L1**

DWord	Bit	MvSize	
		8	32
W2.1	31:0	n/a	MV_Y1_0_L0
W2.2	31:0	n/a	MV_Y1_1_L1
W2.3	31:0	n/a	MV_Y1_1_L0
W2.4	31:0	n/a	MV_Y1_2_L1
W2.5	31:0	n/a	MV_Y1_2_L0
W2.6	31:0	n/a	MV_Y1_3_L0
W2.7	31:0	n/a	MV_Y1_3_L1
W3.0	31:0	n/a	MV_Y2_0_L1
W3.1	31:0	n/a	MV_Y2_0_L0
W3.2	31:0	n/a	MV_Y2_1_L1
W3.3	31:0	n/a	MV_Y2_1_L0
W3.4	31:0	n/a	MV_Y2_2_L1
W3.5	31:0	n/a	MV_Y2_2_L0
W3.6	31:0	n/a	MV_Y2_3_L0
W3.7	31:0	n/a	MV_Y2_3_L1
W4.0	31:0	n/a	MV_Y3_0_L1
W4.1	31:0	n/a	MV_Y3_0_L0
W4.2	31:0	n/a	MV_Y3_1_L1
W4.3	31:0	n/a	MV_Y3_1_L0
W4.4	31:0	n/a	MV_Y3_2_L1
W4.5	31:0	n/a	MV_Y3_2_L0
W4.6	31:0	n/a	MV_Y3_3_L0
W4.7	31:0	n/a	MV_Y3_3_L1

The motion vector(s) for a given sub-macroblock or a sub-partition are uniquely placed in the output message as shown by the non-duplicate fields in *Unpacked Motion Vector Data Block* and *Unpacked Motion Vector Data Block*.

MV_Yx_L0 and MV_Yx_L1 may be present individually or both. If one is not present, the corresponding field must be zero. Subsequently, the duplicated fields will be zero as well.

Motion Vector duplication by sub-macroblocks for a 16x16 macroblock, whereas the 8x8 column is for 4x(8x8) partition without minor shape

DWord	Bit	16x16	16x8	8x16	8x8
W1.0	31:16	**MV_Y0_L1 (A)**	**MV_Y0_L1 (A)**	**MV_Y0_L1**	**MV_Y0_L1**
	15:0	**MV_Y0_L0 (A)**	**MV_Y0_L0 (A)**	**MV_Y0_L0**	**MV_Y0_L0**
W1.1	31:16	**Duplicate (A)**	**Duplicate (A)**	**MV_Y1_L1**	**MV_Y1_L1**
	15:0	**Duplicate (A)**	**Duplicate (A)**	**MV_Y1_L0**	**MV_Y1_L0**
W1.2	31:16	**Duplicate (A)**	**MV_Y2_L1 (B)**	**Duplicate (A)**	**MV_Y2_L1**
	15:0	**Duplicate (A)**	**MV_Y2_L0 (B)**	**Duplicate (A)**	**MV_Y2_L0**
W1.3	31:16	**Duplicate (A)**	**Duplicate (B)**	**Duplicate (B)**	**MV_Y3_L1**
	15:0	**Duplicate (A)**	**Duplicate (B)**	**Duplicate (B)**	**MV_Y3_L0**

Motion Vector duplication by sub-partitions for the first 8x8 sub-macroblock Y0 if any Y0-Y3 contains minor shape (Y1_ to Y3_ have the same format in W2 to W4)

DWord	Bit	8x8	8x4	4x8	4x4
W1.0	31:16	**MV_Y0_L1**	**MV_Y0_0_L1 (A)**	**MV_Y0_0_L1 (A)**	**MV_Y0_0_L1**
	15:0	**MV_Y0_L0**	**MV_Y0_0_L0 (A)**	**MV_Y0_0_L0 (A)**	**MV_Y0_0_L0**
W1.1	31:16	**Duplicate (A)**	**Duplicate (A)**	**MV_Y0_1_L1 (B)**	**MV_Y0_1_L1**
	15:0	**Duplicate (A)**	**Duplicate (A)**	**MV_Y0_1_L0 (B)**	**MV_Y0_1_L0**
W1.2	31:16	**Duplicate (A)**	**MV_Y0_2_L1 (B)**	**Duplicate (A)**	**MV_Y0_2_L1**
	15:0	**Duplicate (A)**	**MV_Y0_2_L0 (B)**	**Duplicate (A)**	**MV_Y0_2_L0**
W1.3	31:16	**Duplicate (A)**	**Duplicate (B)**	**Duplicate (B)**	**MV_Y0_3_L0**
	15:0	**Duplicate (A)**	**Duplicate (B)**	**Duplicate (B)**	**MV_Y0_3_L1**

2.3.1.2.2 Packed-size Motion Vector Data Block

In the packed case, no redundant motion vectors are sent. So the number of motion vectors sent, as specified by **MvQuantity** is the same as the motion vectors that will be packed (**MvPacked**).

The following tables are for information only. Fields like MvQuantity and MvPacked are not required interface fields.

MbSkipFlag	MbType	Description	Mv Quantity	MvSize	(Minimal MvSize)
1	1	P_Skip_16x16	0	8	1
0	1	BP_L0_16x16	1	8	1
0	2	B_L1_16x16	1	8	1
0	3	B_Bi_16x16	2	8	2
0	4	BP_L0_L0_16x8	2	8	4
0	5	BP_L0_L0_8x16	2	8	4
0	6	B_L1_L1_16x8	2	8	8
0	7	B_L1_L1_8x16	2	8	8
0	8	B_L0_L1_16x8	2	8	8
0	9	B_L0_L1_8x16	2	8	8
0	0Ah	B_L1_L0_16x8	2	8	8
0	0Bh	B_L1_L0_8x16	2	8	8
0	0Ch	B_L0_Bi_16x8	3	8	8
0	0Dh	B_L0_Bi_8x16	3	8	8
0	0Eh	B_L1_Bi_16x8	3	8	8
0	0Fh	B_L1_Bi_8x16	3	8	8
0	10h	B_Bi_L0_16x8	3	8	8
0	11h	B_Bi_L0_8x16	3	8	8
0	12h	B_Bi_L1_16x8	3	8	8
0	13h	B_Bi_L1_8x16	3	8	8
0	14h	B_Bi_Bi_16x8	4	8	8
0	15h	B_Bi_Bi_8x16	4	8	8
0	16h	**BP**_8x8	≥4	8 or 32	8 or 32

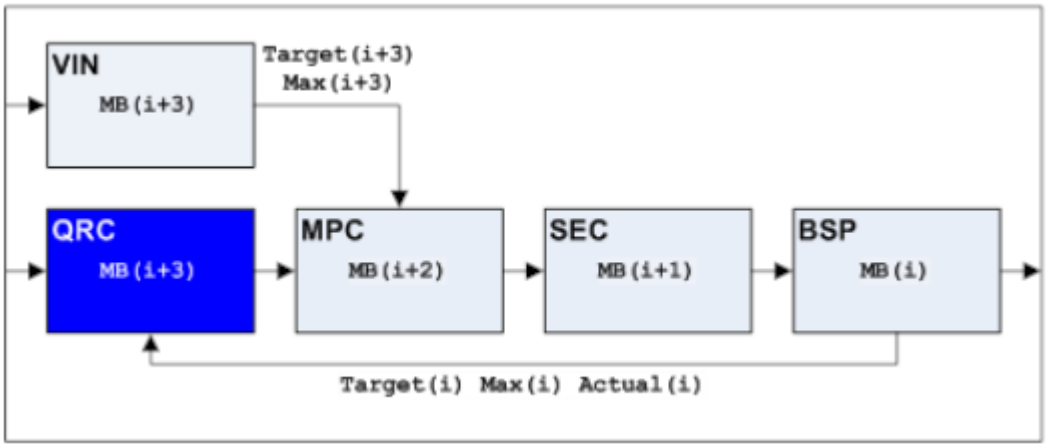
When MbType = 22, BP_8x8, take the sum of four individual 8x8 subblocks

Direct8x8Pattern	SubMb Shape	SubMb PredMode	Description	Mv Quantity	Mv Size	(Min MvSize)
OR	OR	OR		ADD	ADD	ADD
1	0	0	P_Skip_8x8 B_Direct_L0_8x8 (B-Skip_ L0_8x8)	0	2	1
1	0	1	B_Direct_L1_8x8 (B-Skip_ L1_8x8)	0	2	1
1	0	2	B_Direct_Bi_8x8 (B-Skip_ Bi_8x8)	0	2	2
1	3	0	P_Skip_4x4 B_Direct_L0_4x4	0	8	4

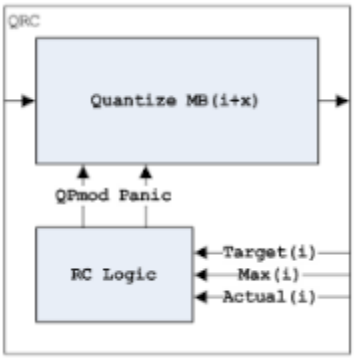
			(B-Skip_ L0_4x4)			
1	3	1	B_Direct_L1_4x4 (B-Skip_ L1_4x4)	0	8	4
1	3	2	B_Direct_Bi_4x4 (B-Skip_ Bi_4x4)	0	8	8
0	0	0	BP_L0_8x8	1	2	1
0	0	1	B_L1_8x8	1	2	1
0	0	2	B_BI_8x8	2	2	2
0	1	0	BP_L0_8x4	2	8	4
0	1	1	B_L1_8x4	2	8	4
0	1	2	B_BI_8x4	4	8	8
0	2	0	BP_L0_4x8	2	8	4
0	2	1	B_L1_4x8	2	8	4
0	2	2	B_BI_4x8	4	8	8
0	3	0	BP_L0_4x4	4	8	4
0	3	1	B_L1_4x4	4	8	4
0	3	2	B_BI_4x4	8	8	8

2.3.1.3 Macroblock Level Rate Control

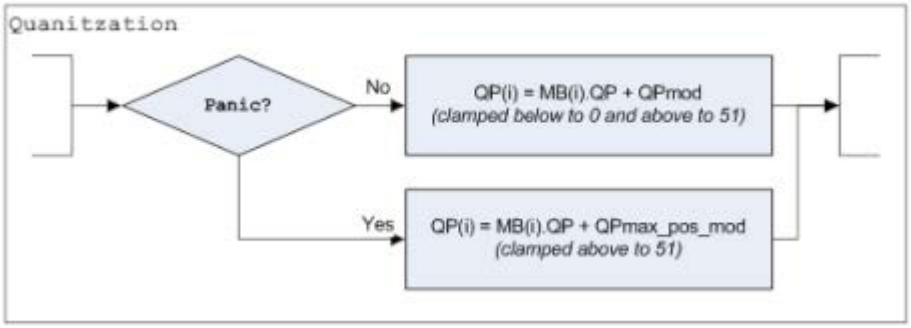
The QRC (Qauntization Rate Control) unit receives data from BSP (Bit Serial Packer) and VIN (Video In) and generates adjustments to QP values across macroblocks.



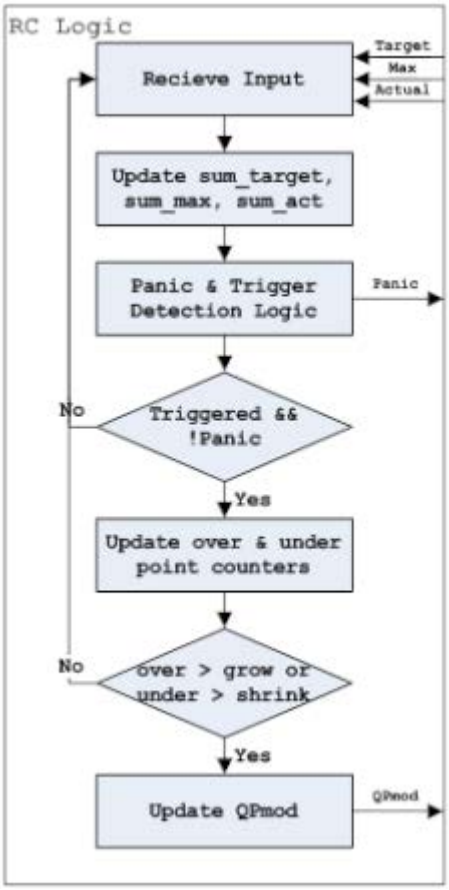
QRC can be logically partitioned into two units as shown below.



Macroblock level rate control is handled by the RC logic and the quantization logic.



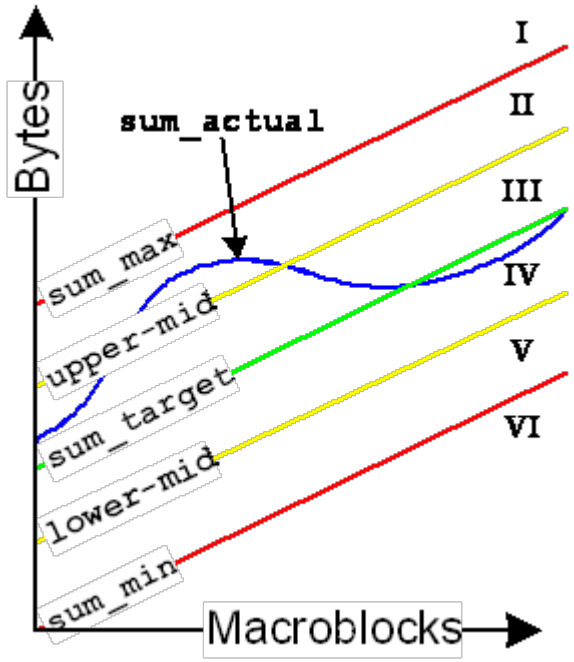
The signals QPmod and panic are generated by the RC logic based on data feedback from BSP. A flowchart of the RC logic is given below.



2.3.1.3.1 Theory of Operation Overview

BSP will generate a byte estimate for each macroblock packed. Additionally, the user will specify a target and max size per macroblock. The running sum of these signals (actual, target, max) creates “curves” which are used to identify when QP adjustments are necessary (see figure below). Three more curves are symmetrically generated by QRC (upper_midpt, lower_midpt, sum_min) from target and max. The values of target and max are specified by the user will dictate the shape of these curves.

The difference between sum_actual and sum_target (called ‘bytediff’) identifies the margin of error between the target and actual sizes. The difference between the current bytediff and the previously calculated bytediff represents the rate of change in this margin over time. The sign of this rate is used to identify if the correction is trending in the appropriate direction (towards bytediff = 0).



QPmod

Each macroblock will have a requested QP (which could vary across macroblocks or remain constant). QPmod is to be added to the QP requested. QPmod will be positive when the target was under-predicted and negative when the target is over-predicted.

QPmod is incremented or decremented when internal counters (called ‘over’ and ‘under’) reach tripping points (called ‘grow’ and ‘shrink’). For each MB processed and based on which region (1-6) sum_actual falls in, various amounts of points are added to either counters. If over exceeds grow, QPmod is incremented whereas if under exceeds shrink, QPmod is decremented.

To dampen the effect of repeated changes in the same direction, an increase in resistance for that direction and decrease in resistance for the complementary direction occurs (called ‘grow_resistance’ and ‘shrink_resistance’). This resistance is added to grow or shrink, which then requires more points to trip the next correction in that direction.

The user can specify guard-bands that limit the amount QPmod can be modified. QPmod cannot exceed QPmax_pos_mod or become less than **-**QPmax_neg_mod_abs.

Triggering

The RC unit begins to modify QPmod occurs only when it is triggered.

Three levels of triggering exist: always, gentle, loose. Always means that RC will be active once sum_actual reaches regions 3 or 4. Gentle will trigger RC once sum_actual reaches regions 2 or 5. Loose waits to trigger RC when sum_actual reaches regions 1 or 6.

RC will deactivate (triggered = false) once sum_actual begins to track sum_target over a series of macroblocks. Specifically, the sign of the rate of change for bytediff is monitored over a window of macroblocks. When the sum of these signs over the window falls within a tolerance value (called 'stable'), triggered will reset to false.

Panic

When enabled, panic mode will occur whenever sum_actual reaches region 1 and will remain so until sum_actual reaches region 4. When panicking, all macroblocks will be quantized with QP = MB(n).QP + QPmax_pos_mod, clamped to 51.

User Controls

This unit achieves a large flexibility by allowing the user to define various key parameters. At the per-macroblock level, the values of target and max are specifed and will create various shapes of curves that sum_actual will be compared against.

Per-slice, the user can specify the triggering sensitivity and the tolerance required to disable the trigger. Additionally, the user can enable panic detection.

The point values assigned to each of the 6 regions are exposed to the user which allow for asymmetrical control for over and under predictions amongst other things. Additionally, the user can specify the initial values of grow and shrink along with the resistance values applied when correction is invoked.

Lastly, the maximum and minimum values for QPmod are also exposed to the user.

2.4 AVC Encoder MBAFF Support

1. **Algorithm**

Prediction of current macroblock motion vector is possible from neighboring macroblocks mbAddrA/mbAddrD/mbAddrB/mbAddrC/mbAddrA+1/mbAddrD+1/mbAddrB+1/mbAddrC+1. The selection of these macroblocks depends on coding type(field/frame) of current macroblock pair and the coding of neighbouring macroblock pair. Following is a generic diagram depicting naming conventions used for neighbouring macroblocks. Selction of these mb pairs desrcibed in detail in following sections.

1.1 Selection of Top LeftMB pair: The selection of Top Left MB pair depends on coding type of current and also top left macroblock pair. Following diagram shows the mapping to be used in MPC unit for the selection of the Top Left MB (D or D+1 macroblock).

1.2 Selection of LeftMB pair The selection of Left MB pair depends on coding type of current and also left macroblock pair. Following diagram shows the mapping to be used in MPC unit for the selection of the Left MB (A or A+1 macroblock).

1.3 Selection of Top MB pair The selection of Top MB pair depends on coding type of current and also top macroblock pair. Following diagram shows the mapping to be used in MPC unit for the selection of the Top MB (B or B+1 macroblock).

1.4 Selection of Top RightMB pair The selection of Top Right MB pair depends on coding type of current and also top right macroblock pair. Following diagram shows the mapping to be used in MPC unit for the selection of the Top Right MB (C or C+1 macroblock).

1.5 Motion Vector and refIdx Scaling Motion vectors and refence index of neighbouring macroblocks (mbAddrA/mbAddrB/mbAddrC/mbAddrD) should be scaled before using them into prediction equations. Again the scaling depends on coding type of current and neighbouring macroblock pair which is described as follows,

- If the current macroblock is a field macroblock and the macroblock mbAddrN is a frame macroblock

mvLXN[1] = mvLXN[1] / 2 (8-214)
refIdxLXN = refIdxLXN * 2 (8-215)

- Otherwise, if the current macroblock is a frame macroblock and the macroblock mbAddrN is a field macroblock

mvLXN[1] = mvLXN[1] * 2 (8-216)
refIdxLXN = refIdxLXN / 2 (8-217)

- Otherwise, the vertical motion vector component mvLXN[1] and the reference index refIdxLXN remain unchanged.

3. MPEG-2

3.1 MPEG2 Common Commands

3.1.1 MFX_MPEG2_PIC_STATE Command

MFX_MPEG2_PIC_STATE		
Source:		VideoCS
Length Bias:		2
This must be the very first command to issue after the surface state, the pipe select and base address setting commands. For MPEG-2 the encoder is called per slice-group, however the picture state is called per picture.Notice that a slice-group is a group of consecutive slices that no non-trivial slice headers are inserted in between.		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_MPEG2_PIC_STATE Format: OpCode
	26:24	**Media Command Opcode** Default Value: 3h MPEG2_COMMON Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 0h Format: OpCode
	15:12	**Reserved** Project: All Format: MBZ
	11:0	**DWord Length** Default Value: 0h Excludes DWord (0,1)= 00Bh, used for normal decode and encode mode000h, a special case to provide a dummy image state for stitch mode operation. In this case, fields in DW1 which is part of the dummy image state command are ignored by hardware. Project: All Format: =n Total Length - 2
1	31:28	**f_code[1][1].** Used for backward motion vector prediction. See ISO/IEC 13818-2 §7.6.3.1 for details
	27:24	**f_code[1][0].** Used for backward motion vector prediction. See ISO/IEC 13818-2 §7.6.3.1 for details

MFX_MPEG2_PIC_STATE

	23:20	**f_code[0][1]** Used for forward motion vector prediction. See ISO/IEC 13818-2 §7.6.3.1 for details
	19:16	**f_code[0][0]** Used for forward motion vector prediction. See ISO/IEC 13818-2 §7.6.3.1 for details
	15:14	**Intra DC Precision** Project: All Format: U32 See ISO/IEC 13818-2 §6.3.10 for details.
	13:12	**Picture Structure** This field specifies whether the picture is encoded in the form of a frame picture or one field (top or bottom) picture. See ISO/IEC 13818-2 §6.3.10 for details.Format = MPEG_PICTURE_STRUCTURE00 = Reserved01 = MPEG_TOP_FIELD10 = MPEG_BOTTOM_FIELD11 = MPEG_FRAME
	11	**TFF (Top Field First)** When two fields are stored in a picture, this bit indicates if the top field is the first field.For a frame P picture, the value 1 indicates that the top field of the reconstructed frame is the first field output by the decoding process, the same as defined in ISO/IEC 13818-2 §6.3.10. Particularly, it is used by the hardware to calculate derivative motion vectors from the dual-prime motion vectors.For a field P picture, hardware uses this bit together with the Picture Structure to determine if the current picture is the Second Field. In this case, the definition of this bit differs from ISO/IEC 13818-2 §6.3.10 – software must derive the value for this bit according to the following relation:Picture Structure = top fieldPicture Structure = bottom fieldSecond Field = 0TFF = 1TFF = 0Second Field = 1TFF = 0TFF = 1
	10	**Frame Prediction Frame DCT** This field provides constraints on the DCT type and prediction type. It affects the syntax of the bitstream.
	9	**Concealment Motion Vector Flag** This field indicates if the concealment motion vectors are coded in intra macroblocks. It affects the syntax of the bitstream.
	8	**Quantizer Scale Type** Format: MPEG_Q_SCALE_TYPE This field specifies the quantizer scaling type. Value / Name / Description 0h / / MPEG_QSCALE_LINEAR 1h / / D MPEG_QSCALE_NONLINEAR esc
	7	**Intra VLC Format** This field is used by VLD
	6	**Scan Order** Format: MPEG_INVERSESCAN_TYPE This field specifies the Inverse Scan method for the DCT-domain coefficients in the blocks of the current picture. Value / Name / Description 0h / / MPEG_ZIGZAG_SCAN 1h / / MPEG_ALTERNATE_VERTICAL_SCAN
	5:0	**Reserved**
2	31:24	**Reserved** Format: MBZ

MFX_MPEG2_PIC_STATE

	23:15	**Reserved** Format: MBZ
	14	**LoadSlicePointerFlag – LoadBitStreamPointerPerSlice** Exists If: Encoder To support multiple slice picture and additional header/data insertion before and after an encoded slice.When this field is set to 0, bitstream pointer is only loaded once for the first slice of a frame. For subsequent slices in the frame, bitstream data are stitched together to form a single output data stream.When this field is set to 1, bitstream pointer is loaded for each slice of a frame. Basically bitstream data for different slices of a frame will be written to different memory locations. **Value** / **Name** / **Description** 0h — Load BitStream Pointer only once for the first slice of a frame 1h — Load/reload BitStream Pointer only once for the each slice, reload the start location of the bitstream buffer from the Indirect PAK-BSE Object Data Start Address field
	13	**Reserved** Format: MBZ
	12	**Reserved** Format: MBZ
	11	**Reserved** Format: MBZ
	10:9	**Picture Coding Type** Format: MPEG_PICTURE_CODING_TYPE This field identifies whether the picture is an intra-coded picture (I), predictive-coded picture (P) or bi-directionally predictive-coded picture (B). See ISO/IEC 13818-2 §6.3.9 for details. **Value** / **Name** 00b — Reserved 01b — MPEG_I_PICTURE 10b — 10 = MPEG_P_PICTURE 11b — MPEG_B_PICTURE
	8:2	**Reserved** Format: MBZ
	1	**MismatchControlDisabled** These 2 bits flag disables mismatch control of the inverse transformation for some specific cases during reference reconstruction. **Value** / **Name** / **Description** 00b — Mismatch control applies to all MBs 01b — Disable mismatch control to all intra MBs whose all AC-coefficients are zero. 10b — Disable mismatch control to all MBs whose all AC-coefficients are zero. 11b — Disable mismatch control to all MBs.
	0	**Disable Mismatch** To disable MPEG2 IDCT fixed point arithmetic correction
3	31	**Reserved** Format: MBZ
	30:29	**Reserved** Format: MBZ
	23:16	**FrameHeightInMBsMinus1[7:0] (Picture Height in Macroblocks)** Format: U8 To support MB error concealment.

MFX_MPEG2_PIC_STATE

	15:8	**Reserved** Format: MBZ for future supporting width > 4K
	7:0	**FrameWidthInMBsMinus1[7:0] (Picture Width in Macroblocks)** Project: All Format: U8 To support MB error concealment.
4	31:16	**MinFrameWSize** Project: All Format: U32 Format: GraphicsAddress[31:0]U32 – Minimum Frame Size [15:0] (16-bit) (Encoder Only)Mininum Frame Size is specified to compensate for intel Rate ControlCurrently zero fill (no need to perform emulation byte insertion) is done only to the end of the CABAC_ZERO_WORD insertion (if any) at the last slice of a picture. Intel encoder parameter, not part of DXVA. The caller should always make sure that the value, represented by Mininum Frame Size, is always less than maximum frame size FrameBitRateMax (DWORD 10 bits 29:16). This field is reserved in Decode mode.

Value	Name	Description
[0,0003FFFFh]		The programmable range when MinFrameWSizeUnits is 00.
[0,000FFFFFh]		The Programmable range when MinFrameWSizeUnits is 01.
[0,03FFFFFFh]		The Programmable range when MinFrameWSizeUnits is 10.
[0,FFFFFFFFh]		The Programmable range when MinFrameWSizeUnits is 11.
0h	**[Default]**	

	15	**Reserved** Project: All Format: MBZ
	14:12	**RoundInterAC,** rounding precision for non-Intra AC000: +1/16001: +2/16010: +3/16011: +4/16100: +5/16101: +6/16110: +7/16111: +8/16
	11	**Reserved** Format: MBZ
	10:8	**RoundIntraAC** Project: All Format: U32 rounding precision for Intra AC000: +1/16001: +2/16010: +3/16011: +4/16100: +5/16101: +6/16110: +7/16111: +8/16
	7	**Reserved** Format: MBZ
	6:4	**RoundInterDC** rounding Precision for non-Intra-DC000: +1/16001: +2/16010: +3/16011: +4/16100: +5/16101: +6/16110: +7/16111: +8/16
	3	**Reserved** Format: MBZ
	2:1	**RoundIntraDC**

MFX_MPEG2_PIC_STATE

		rounding Precision for Intra-DC00: +1/801: +2/810: +3/811: +4/8
	0	**Reserved**
5	31:17	**Reserved(for future Mask bits)**
	16	**FrameSizeControlMask** Frame size conformance maskThis field is used when MacroblockStatEnable is set to 1. **Value** / **Name** / **Description** 0h / / Do not change Slice Quantization Parameter values in MFC_MPEG2_SLICEGROUP_STATE with suggested slice QP value for frame level Rate control 1h / / Replace Slice Quantization Parameter values in MFC_MPEG2_SLICEGROUP_STATE with suggested slice QP value for frame level Rate control values in MFC_IMAGE_STATUS control register.
	15:13	**Reserved**
	12	**InterMBForceCBPZeroControlMask** Format: U32 Inter MB Force CBP ZERO mask. **Value** / **Name** / **Description** / **Project** [0,FFFFFFFFh] / / / 0h / / No effect / All 1h / / Zero out all A/C coefficients for the inter MB violating Inter Confirmance / All
	11:10	**MinFrameWSizeUnits** This field is the Minimum Frame Size Units **Value** / **Name** / **Description** / **Project** 00b / compatibility mode / Minimum Frame Size is in old mode (words, 2bytes) / All 01b / 16 byte / Minimum Frame Size is in 16bytes / All 10b / 4Kb / Minimum Frame Size is in 4Kbytes / All 11b / 16Kb / Minimum Frame Size is in 16Kbytes / All
	9	**MBRateControlMask** MB Rate Control conformance maskThis field is ignored when MacroblockStatEnable is disabled or MB level Rate control flag for the current MB is disable in Macroblock Status Buffer. **Value** / **Name** / **Description** 0h / / Do not change QP values of inter macroblock with suggested QP values in Macroblock Status Buffer 1h / / Apply RC QP delta for all macroblock
	8	**Reserved**
	7	**Reserved** Format: MBZ
	6:4	**Reserved**
	3	**FrameBitRateMinReportMask** This is a mask bit controlling if the condition of frame level bit count is less than FrameBitRateMin. **Value** / **Name** / **Description** / **Project** 0h / Disable / Do not update bit0 of MFC_IMAGE_STATUS control register. / All 1h / Enable / set bit0 and bit 1of MFC_IMAGE_STATUS control register if the total frame level bit counter is less than or equal to Frame Bit rate Minimum limit. / All
	2	**FrameBitRateMaxReportMask** This is a mask bit controlling if the condition of frame level bit count exceeds FrameBitRateMax. **Value** / **Name** / **Description** / **Project** 0h / Disable / Do not update bit0 of MFC_IMAGE_STATUS control register. / All

		1h	Enable	set bit0 and bit 1 of MFC_IMAGE_STATUS control register if the total frame level bit counter is greater than or equal to Frame Bit rate Maximum limit.	All

1 InterMBMaxSizeReportMask

This is a mask bit controlling if the condition of any inter MB in the frame exceeds InterMBMaxSize.

Value	Name	Description
0h		Do not update bit0 of MFC_IMAGE_STATUS control register.
1h		set bit0 of MFC_IMAGE_STATUS control register if the total bit counter for the current MB is greater than the Inter MB Conformance Max size limit.

0 IntraMBMaxSizeReportMask

This is a mask bit controlling if the condition of any intra MB in the frame exceeds IntraMBMaxSize.

Value	Name	Description	Project
0h		Do not update bit0 of MFC_IMAGE_STATUS control register.	All
1h		set bit0 of MFC_IMAGE_STATUS control register if the total bit counter for the current MB is greater than the Intra MB Conformance Max size limit.	All

6 (Encode only) 31:28 Reserved

Project:	All
Format:	MBZ

27:16 InterMBMaxSize

Default Value:	FFFh

This field, Inter MB Conformance Max size limit,indicates the allowed max bit count size for Inter MB

15:12 Reserved

Project:	All
Format:	MBZ

11:0 IntraMBMaxSize

Default Value:	FFFh

This field, Intra MB Conformance Max size limit,indicates the allowed max bit count size for Intra MB

7 31:0 Reserved

Project:	All
Format:	MBZ

8 (Encode only) 31:24 SliceDeltaQPMax[3]

Format:	S7

This field is the Slice level delta QP for total bit-count above FrameBitRateMax - first 1/8 regionThis field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame exceeds FrameBitRateMax but is within 1/8 of FrameBitRateMaxDelta above FrameBitRateMax, i.e., in the range of (FrameBitRateMax, (FrameBitRateMax+ FrameBitRateMaxDelta>>3).

Range: [-30,30]

Value	Name	Project
0h	Disable	All
1h	Enable	All

Errata	Description
#	

MFX_MPEG2_PIC_STATE

	23:16	**SliceDeltaQPMax[2]** Format: S7 Range: [-30,30] This field is the Slice level delta QP for bit-count above FrameBitRateMax - above 1/8 and below 1/ 4 This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between 1/8 and ¼ of FrameBitRateMaxDelta above FrameBitRateMax, i.e., in the range of ((FrameBitRateMax+ FrameBitRateMaxDelta>>3), (FrameBitRateMax+ FrameBitRateMaxDelta>>2).
	15:8	**SliceDeltaQPMax[1]** Format: S7 Range: [-30,30] This field is the Slice level delta QP for bit-count above FrameBitRateMax – above1/ 4 and below 1/2 This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between ¼ and ½ of FrameBitRateMaxDelta above FrameBitRateMax, i.e., in the range of ((FrameBitRateMax+ FrameBitRateMaxDelta>>2), (FrameBitRateMax+ FrameBitRateMaxDelta>>1).
	7:0	**SliceDeltaQPMax[0]** Format: S7 Range: [-30,30] This field is the Slice level delta QP for bit-count above FrameBitRateMax - above 1/ 2This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is above FrameBitRateMax by more than half the distance of FrameBitRateMaxDelta , i.e., in the range of ((FrameBitRateMax+ FrameBitRateMaxDelta>>1), infinite).
9 (Encode only)	31:24	**SliceDeltaQPMin[3]** Format: S7 Range: [-30,30] This field is the Slice level delta QP for total bit-count below FrameBitRateMin - first 1/8 regionThis field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is less than FrameBitRateMin and greater than or equal to 1/8 the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of [(FrameBitRateMin- FrameBitRateMinDelta>>3), FrameBitRateMin).
	23:16	**SliceDeltaQPMin[2]** Format: S7 Range: [-30,30] This field is the Slice level delta QP for bit-count below FrameBitRateMin – below 1/ 8 and above 1/ 4This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between one-eighth and quarter the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of [(FrameBitRateMin- FrameBitRateMinDelta>>2), (FrameBitRateMin- FrameBitRateMinDelta>>3)).

MFX_MPEG2_PIC_STATE

	15:8	**SliceDeltaQPMin[1]** Format: S7 Range: [-30,30] This field is the Slice level delta QP for bit-count below FrameBitRateMin– below 1/4 and above 1/ 2This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between quarter and half the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of [(FrameBitRateMin-FrameBitRateMinDelta>>1), (FrameBitRateMin- FrameBitRateMinDelta>>2)).
	7:0	**SliceDeltaQPMin[0]** Format: S7 Range: [-30,30] This field is the Slice Level Delta QP for bit-count below FrameBitRateMin – below 1/ 2This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is below FrameBitRateMin by more than half the distance of FrameBitRateMinDelta , i.e., in the range of [0, (FrameBitRateMin- FrameBitRateMinDelta>>1).
10 (Encode only)	31	**FrameBitrateMaxUnit** This field is the Frame Bitrate Maximum Limit Units.

Value	Name	Description	Project
0h	Byte	FrameBitRateMax is in units of 32 Bytes when FrameBitrateMaxUnitMode is 1 and in units of 128 Bytes if FrameBitrateMaxUnitMode is 0	All
1h	Kilobyte	FrameBitRateMax is in units of 4KBytes Bytes when FrameBitrateMaxUnitMode is 1 and in units of 16KBytes if FrameBitrateMaxUnitMode is 0	All

30 **FrameBitrateMaxUnitMode**
BitFiel This field is the Frame Bitrate Maximum Limit Units.dDesc

Value	Name	Description	Project
0h	Compatibility mode	FrameBitRateMaxUnit is in old mode (128b/16Kb)	All
1h	New mode	FrameBitRateMaxUnit is in new mode (32byte/4Kb)	All

29:16 **FrameBitRateMax**
This field is the Frame Bitrate Maximum Limit. This field along with FrameBitrateMaxUnit determines maximum allowed bits in a frame before multi-pass gets triggered (when enabled). In other words, multi-pass is triggered when the actual frame byte count exceeds this value. When FrameBitrateMaxUnitMode is 0(compatibility mode) bits 16:27 should be used, bits 28 and 29 should be 0.

Value	Name	Description
0-512KB		The programmable range 0-512KB when FrameBitrateMaxUnit is 0.
0-8190KB		The programmable range 0-8190KB when FrameBitrateMaxUnit is 1.

15 **FrameBitrateMinUnit**
This field is the Frame Bitrate Minimum Limit Units.

Value	Name	Description	Project
0h	Byte	FrameBitRateMax is in units of 32 Bytes when FrameBitrateMinUnitMode is 1 and in units of 128 Bytes if FrameBitrateMinUnitMode is 0	All
1h	KiloByte	FrameBitRateMax is in units of 4KBytes Bytes when FrameBitrateMaxUnitMode is 1 and in units of 16KBytes if FrameBitrateMaxUnitMode is 0	All

14 **FrameBitrateMinUnitMode**

This field is the Frame Bitrate Minimum Limit Units.ValueNameDescriptionProject

Value	Name	Description	Project
0h	compatibility mode	FrameBitRateMaxUnit is in old mode (128b/16Kb)	All
1h	New Mode	FrameBitRateMaxUnit is in new mode (32byte/4Kb)	All

13:0 **FrameBitRateMin**

This field is the Frame Bitrate Minimum Limit ()This field along with FrameBitrateMinUnit determines minimum allowed bits in a Frame before Multi-Pass gets triggered (when enabled). In other words, multi-pass is triggered when the actual frame byte count is less than this value. When FrameBitrateMinUnitMode is 0 (compatibility mode) bits 0:11 should be used, bits 12 and 13 should be 0. Range: The programmable range 0-512KB When FrameBitrateMinUnit is in 0. Programmable range is 0–8190 KB when FrameBitrateMinUnit is in 1

11 (Encode only)

31 **Reserved**

Format:	MBZ

30:16 **FrameBitRateMaxDelta**

Default Value:	0h
Project:	All
Access:	None
Exists If:	Always
Format:	U32
Format:	GraphicsAddress[31:0]U32

This field is used to select the slice delta QP when FrameBitRateMax Is exceeded. It shares the same FrameBitrateMaxUnit.
The programmable range is either 0- 512KB or 4MBB in FrameBitrateMaxUnit of 128 Bytes or 16KB respectively.

This field is used to select the slice delta QP when FrameBitRateMax Is exceeded. It shares the same FrameBitrateMaxUnit. When FrameBitrateMaxUnitMode is 0(compatibility mode) bits 16:27 should be used, bits 28, 29 and 30 should be 0.

15 **Reserved**

Project:	All
Format:	MBZ

14:0 **FrameBitRateMinDelta**

This field is used to select the slice delta QP when FrameBitRateMin Is exceeded. It shares the same FrameBitrateMinUnit. When FrameBitrateMinUnitMode is 0(compatibility mode) bits 0:11 should be used, bits 12, 13 and 14 should be 0.Note: HW requires the following condition FrameBitRateMinDelta <= 2*FrameBitRateMinMust be true, otherwise it may cause unpredicted behavior.

Value	Name	Description
0-1024KB		The programmable range 0-1024KB When FrameBitrateMinUnit is in 32Bytes.
0-16380KB		Programmable range is 0–16380KB when FrameBitrateMinUnit is in 4Kbytes.

12 31:0 **Reserved**

Format:	MBZ

3.2 MPEG2 Decoder Commands

3.2.1 MFD_MPEG2_BSD_OBJECT Command (pipeline)

MFD_MPEG2_BSD_OBJECT			
Source:		VideoCS	
Length Bias:		2	
Different from AVC and VC1, MFD_MPEG2_BSD_OBJECT command is pipelinable. This is for performance purpose as in MPEG2 a slice is defined as a group of MBs of any size that must be within a macroblock row.Slice header parameters are passed in as inline data and the bitstream data for the slice is passed in as indirect data. Of the inline data, slice_horizontal_position and slice_vertical_position determines the location within the destination picture of the first macroblock in the slice. The content in this command is identical to that in the MEDIA_OBJECT command in VLD mode described in the Media Chapter.			
DWord	**Bit**	**Description**	
0	31:29	**Command Type**	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	**Pipeline**	
		Default Value:	2h MFD_MPEG2_BSD_OBJECT
		Format:	OpCode
	26:24	**Media Command Opcode**	
		Default Value:	3h MPEG2_DEC
		Format:	OpCode
	23:21	**SubOpcode A**	
		Default Value:	1h
		Format:	OpCode
	20:16	**SubOpcode B**	
		Default Value:	8h
		Format:	OpCode
	15:12	**Reserved**	
		Project:	All
		Format:	MBZ
	11:0	**DWord Length**	
		Default Value:	0003h Excludes DWord (0,1)
		Project:	All
		Format:	=n Total Length - 2
1	31:0	**Indirect BSD Data Length**	
		Project:	All
		Format:	U32
		It is the length in bytes of the bitstream data for the current slice. It includes the first byte of the first macroblock and the last non-zero byte of the last macroblock in the slice. Specifically, the zero-padding bytes (if present) and the next start-code are excluded. This field is sized to support beyond MPEG-2 MP@HL bitstream (<4K). According to Table 8-6 of ISO/IEC 13818-2, the maximum number of bits per macroblock for 4:2:0 is 4608. So the maximum slice size for 4K x 4K is 4608 * 256 / 8 = 147,456 bytes (0x24000), which requires 18 bits.	

MFD_MPEG2_BSD_OBJECT			
		Programming Notes	Project
		As MPEG-2 spec does not post any limitation of the size of zero-padding bytes, it is possible to have a slice data with large length (including zero-padding bytes). As the data beyond 0x10E00 would only be zero bytes for a valid slice data	
		Hardware does not handle zero-padding at the end of the slice data so driver needs to program the datalength from the first byte of the first macroblock and the last non-zero byte of the last macroblock in the slice. This datalength must exclude all the extra zero padding at the end of a slice bitstream.	
		Bits [31:24] must be programmed to 0.	
2	31:29	**Reserved** Project: All Format: MBZ	
	28:0	**Indirect Data Start Address** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This pointer is relative to the BSD Indirect Object Base Address.Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the MPEG2 VLD bitstream data This address points to the first byte of the MB layer data, i.e. not including slice header.	
3..4	31:0	**Inline Data** All the required Slice Header parameters and error handling settings are captured as inline data of the MPEG2_BSD_OBJECT command. It has a fixed size of 2 DWs. Its definition is described in the next section.	

3.2.1.1 Inline Data Description in MFD_MPEG2_BSD_OBJECT

Inline Data Description in MFD_MPEG2_BSD_OBJECT		
Source:	VideoCS	
Default Value:	0x00000000, 0x00000000	
DWord	**Bit**	**Description**
3	31	**Reserved** Format: MBZ
	30:24	**Slice Horizontal Position** Format: U7 in Macroblocks This field indicates the horizontal position (in macroblock units) of the first macroblock in the slice.
	23	**Reserved** Format: MBZ
	22:16	**Slice Vertical Position** Format: U7 in Macroblocks This field indicates the vertical position (in macroblock units) of the first macroblock in the slice.
	15	**Reserved**

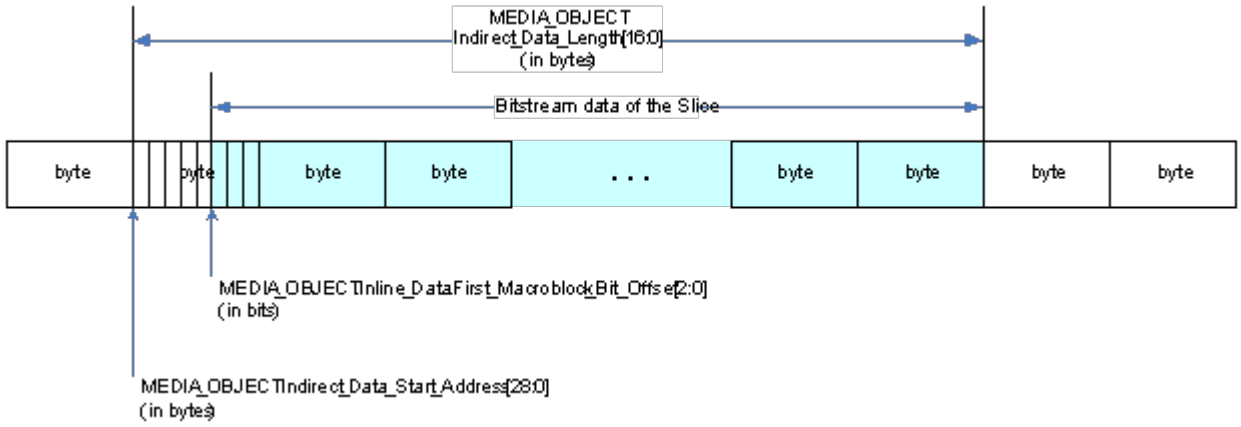
Inline Data Description in MFD_MPEG2_BSD_OBJECT				
		Format:	MBZ	
	14:8	**Macroblock Count**		
		Format:	U7 in Macroblocks	
		This field indicates the number of macroblocks in the slice, including skipped macroblocks.		
	7:6	**Reserved**		
		Format:	MBZ	
	5	**Last Pic Slice** This bit is added to support error concealment at the end of a picture.		
		Value / **Name** / **Description**		
		1h		The current Slice is the last Slice of the entire picture
		0h		The current Slice is not the last Slice of current picture
	3	**Is Last MB**		
		Value / **Name** / **Description**		
		1h		The current MB is the last MB in the current Slice
		0h		The current MB is not the last MB in the current Slice
	2:0	**First Macroblock Bit Offset**		
		Format:	U3	
		This field provides the bit offset of the first macroblock in the first byte of the input bitstream.		
4	31:29	**Reserved**		
		Format:	MBZ	
	28:24	**Quantizer Scale Code**		
		Format:	U5	
		This field sets the quantizer scale code of the inverse quantizer. It remains in effect until changed by a decoded quantizer scale code in a macroblock. This field is decoded from the slice header by host software.		
	23:0	**Reserved**		
		Format:	MBZ	

3.2.1.2 Indirect Data Description

The indirect data start address in MFD_MPEG2_BSD_OBJECT specifies the starting Graphics Memory address of the bitstream data that follows the slice header. It provides the byte address for the first macroblock of the slice. Together with the First Macroblock Bit Offset field in the inline data, it provides the bit location of the macroblock within the compressed bitstream.

The indirect data length in MFD_MPEG2_BSD_OBJECT provides the length in bytes of the bitstream data for this slice. It includes the first byte of the first macroblock and the last **non-zero** byte of the last macroblock in the slice. Specifically, the zero-padding bytes (if present) and the next start-code are excluded. Hardware ignores the contents after the last non-zero byte. *Indirect Data Description* illustrates these parameters for a slice data.

Indirect data buffer for a slice



4. JPEG

4.1 JPEG Decoder Commands

4.1.1 MFD_JPEG_BSD_OBJECT Command

MFD_JPEG_BSD_OBJECT		
Project:		All
Source:		VideoCS
Length Bias:		2
Decoder		

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFD_JPEG_BSD_OBJECT Format: OpCode
	26:24	**Media Command Opcode** Default Value: 7h JPEG_DEC Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h Format: OpCode
	20:16	**SubOpcode B** Default Value: 8h Format: OpCode
	15:12	**Reserved** Project: All Format: MBZ
	11:0	**DWord Length** Default Value: 004h Excludes DWord (0,1) Project: All Format: =n Total Length - 2
1	31:0	**Indirect Data Length** Project: All . It is the length in bytes of the bitstream data for the current Scan. It includes the first byte of the first MCU and the last non-zero byte of the last MCU in the Scan. Specifically, the zero-padding bytes (if present) are excluded. Hardware ignores the contents after the last non-zero byte.
2	31:29	**Reserved** Project: All

MFD_JPEG_BSD_OBJECT

		Format: MBZ
	28:0	**Indirect Data Start Address** Project: All This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This pointer is relative to the BSD Indirect Object Base Address.Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the JPEG bitstream data
3	31:29	**Reserved** Project: All Format: MBZ
	28:16	**Scan Horizontal Position** Project: All Format: U13 bits in blocks This field indicates the horizontal position (in block units) of the first MCU in the Scan.
	15:13	**Reserved** Project: All Format: U13 bits in blocks
	12:0	**Scan Vertical Position** Project: All Format: U13 bits in blocks This field indicates the vertical position (in block units) of the first MCU in the Scan.
4	31	**Reserved** Format: MBZ
	30	**Interleaved** Value / Name / Description 0 / Non-Interleaved / one component in the Scan 1 / Interleaved / multiple components in the Scan
	29:27	**Scan Components** Bit0: Y Bit1: U Bit2: V For example, if non-interleaved Y, then it will be set to 001b. If interleaved Y, U, and V, it will be set to 111b.
	26	**Reserved** Format: MBZ
	25:0	**MCU Count** Project: All Format: U26 This field indicates the number of MCUs in the Scan.
5	31:16	**Reserved**

MFD_JPEG_BSD_OBJECT		
		Project: All Format: MBZ
	15:0	**RestartInterval(16 bit)** Project: All Format: U32 Specifies the number of MCU in restart interval. Valid values are 1->0xFFFFValue of 0 implies that all the SCAN have only one ECS.

4.1.2 MFX_JPEG_PIC_STATE Decoder

MFX_JPEG_PIC_STATE_Decoder Only		
Source:		VideoCS
Length Bias:		2
Exists If:		Decoder Only
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_MULTI_DW Format: OpCode
	26:24	**Media Command Opcode** Default Value: 7h JPEG_COMMON Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 0h Format: OpCode
	15:12	**Reserved** Project: All Format: MBZ
	11:0	**DWord Length** Project: All Format: =n Total Length - 2 Value / Name / Description 0001h / **[Default]** / Excludes DWord (0,1)
1	31:21	**Reserved**

MFX_JPEG_PIC_STATE_Decoder Only

	Format:	MBZ

20:19 Reserved

Format:	MBZ

18 Reserved

Format:	MBZ

17:16 Reserved

15:12 Reserved

Format:	MBZ

11:8 Reserved

Format:	MBZ

7:6 Reserved

Format:	MBZ

5:4 Rotation

Value	Name	Description
00b		no rotation
01b		rotate clockwise 90 degree
10b		rotate counter-clockwise 90 degree (same as rotating 270 degree clockwise)
11b		rotate 180 degree (NOT the same as flipped on the x-axis)

3 Reserved

Format:	MBZ

2:0 Input Format YUV

Exists If:	Always
Format:	U32 GraphicsAddress[31:0]

Value	Name	Description
0	**[Default]**	YUV400 (grayscale image)
1		YUV420
2		YUV422H_2Y (Horizontally chroma 2:1 subsampled) – horizontal 2 Y-block, 1U and 1V
3		YUV444
4		YUV411
5		YUV422V_2Y (Vertically chroma 2:1 subsampled) – vertical 2 Y-blocks, 1U and 1V
6		YUV422H_4Y - 2x2 Y-blocks, vertical 2U and 2V
7		YUV422V_4Y - 2x2 Y-blocks, horizontal 2U and 2V

2 31:29 Reserved

Format:	MBZ

28:16 Frame Height In Blocks Minus 1

Format:	U32

(The number of blocks in height) – 1.
This value is calculated using the number of lines Y and vertical sampling factor of the first component V_1 in Frame header. See the note following this table.
For interleaved components, (((Y + (V_1*8 -1)) / (V_1*8)) * V_1) – 1

MFX_JPEG_PIC_STATE_Decoder Only		
		For non-interleaved components, ((Y + 7) / 8) – 1.
	15:13	**Reserved** Format: MBZ
	12:0	**Frame Width In Blocks Minus 1** Format: U32 (The number of blocks in width) – 1. This value is calculated using the number of samples per line X and horizontal sampling factor of the first component H_1 in Frame header. See the note following this table. For interleaved components, (((X + (H_1 *8 -1)) / (H_1 *8)) * H_1) – 1. For non-interleaved components, ((X + 7) / 8) – 1.

For JPEG decoding, the following program note is informative.

For **Rotation**, it is important to note that rotation of 90 or 270 degrees also requires exchanging **FrameWidthInBlksMinus1** with **FrameHeightInBlksMinus1** in the command. In addition, the rotation of 90 or 270 degrees also requires transportation of the quantization matrix will be transposed into the position (y, x).

Chroma type is determined by the values of horizontal and vertical sampling factors of the components (*Hi* and Vi where *i* is a component id) in the Frame header as shown in the following table.

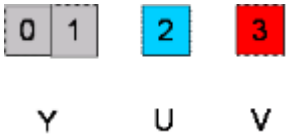
	H1	*H2*	*H3*	*V1*	*V2*	*V3*
0: YUV400	r	Not available	Not available	r	Not available	Not available
1: YUV420	2	1	1	2	1	1
2:YUV422H_2Y	2	1	1	1	1	1
3: YUV444	1	1	1	1	1	1
4: YUV411	4	1	1	1	1	1
5: YUV422V_2Y	1	1	1	2	1	1
6: YUV422H_4Y	2	1	1	2	2	2
7: YUV422V_4Y	2	2	2	2	1	1

For YUV400, the value of *V1* can be 1, 2, or 3 and will be same as the value of *H1*, and the Minimum coded unit (MCU) is one 8x8 block. For the other chroma formats, if non-interleaved data, the MCU is one

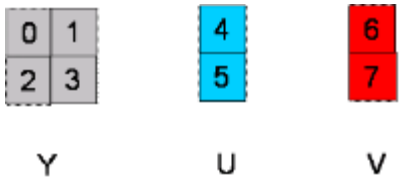
8x8 block. For interleaved data, the MCU is the sequence of block units defined by the sampling factors of the components.

For example, the following figures show the MCU structures of interleaved data and the decoding order of blocks in the MCU:

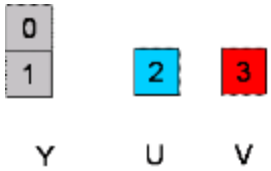
422H_2Y



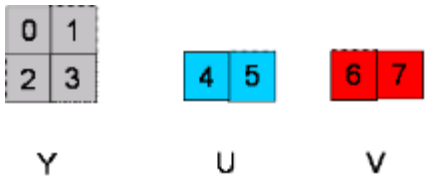
422H_4Y



422V_2Y



422V_4Y



If picture width *X* in the Frame header is not a multiple of 8, the decoding process needs to extend the number of column to complete the right-most sample blocks. If the component is to be interleaved, the decoding process needs to extend the number of samples by one or more additional blocks so that the number of blocks is an integer multiple of *Hi*. In other words, “The number of blocks in width” in the table should be an integer multiple of (8x*H1*). Similarly, if picture height *Y* in the Frame header is not a multiple of 8, the decoding process needs extend the number of lines to complete bottom-most block-row. If the component is to be interleaved, the decoding process also needs to extend the number of lines by one or more additional block-rows so that the number of block-row is an integer multiple of (8x*V1*). For example, if non-interleaved YUV411 with *X*=270, then “The number of blocks in width” shall be (270 + 7) / 8 = 34, where “/” is integer division. Therefore, **FrameWidthInBlksMinus1** will be set to 33. However, for interleaved data, “The number of blocks in width” shall be ((270 + 31) / 32) x 4 = 36. Therefore, **FrameWidthInBlksMinus1** will be set to 35.

4.1.3 MFX_JPEG_HUFF_TABLE_STATE

MFX_JPEG_HUFF_TABLE_STATE	
Project:	All
Source:	VideoCS
Length Bias:	2

This Huffman table commands contains both DC and AC tables for either luma or chroma. Once a Huffman table has been defined for a particular destination, it replaces the previous tables stored in that destination and shall be used in the remaining Scans of the current image. A Huffman table will be sent to H/W only when it is loaded from bitstream.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_MULTI_DW Format: OpCode
	26:24	**Media Command Opcode** Default Value: 7h JPEG_COMMON Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 2h Format: OpCode
	15:12	**Reserved** Project: All Format: MBZ
	11:0	**DWord Length** Default Value: 033Dh Excludes DWord (0,1) Project: All Format: =n Total Length - 2
1	31:1	**Reserved** Format: MBZ
	0	**HuffTableID (1-bit)** Identifies the huffman table. Value: 0 — Name: Y — Description: Huffman table for Y
2..4	31:0	**DC_BITS (12 8-bit array)** The number of DC Huffman codes of length i, where i is 1~12
5..7	31:0	**DC_HUFFVAL (12 8-bit array)** The value associated with each DC Huffman code of length i.
8..11	31:0	**AC_BITS (16 8-bit array)** the list of Li, number of Huffman codes of length i, where i is 1~16

MFX_JPEG_HUFF_TABLE_STATE		
12..51	31:0	**AC_HUFFVAL (160 8-bit array)** the list of Vi,j, the value associated with each Huffman code of length i
52	31:16	**Reserved** Project: All Format: MBZ
	15:0	**AC_HUFFVAL(2-8 bit array)** In AC table, BITS can have up to 16-bit codeword. Li can be 0 ~ 162. HUFFVAL will have a list of likely random distributed values

5. More Decoder and Encoder

5.1 MFD IT Mode Decode Commands

5.1.1 MFD_IT_OBJECT Command

MFD_IT_OBJECT		
Project:		All
Source:		VideoCS
Length Bias:		2
All weight mode (default and implicit) are mapped to explicit mode. But the weights come in either as explicit or implicit.		
DWord	**Bit**	**Description**
0	31:29	**Command Type**
		Default Value: 3h PARALLEL_VIDEO_PIPE
		Format: OpCode
	28:27	**Pipeline**
		Default Value: 2h MFD_IT_OBJECT
		Format: OpCode
	26:24	**Media Command Opcode**
		Default Value: 0h MFX_COMMON_DEC
		Format: OpCode
	23:21	**SubOpcode A**
		Default Value: 1h
		Format: OpCode
	20:16	**SubOpcode B**
		Default Value: 9h
		Format: OpCode
	15:12	**Reserved**
		Project: All
		Format: MBZ
	11:0	**DWord Length**
		Default Value: 0h Excludes DWord (0,1) For AVC = Ch
		Project: All
		Format: =n Total Length – 2Note: Regardless of the mode, inline data must be present in this command.
1	31:10	**Reserved**
		Project: All
		Format: MBZ
	9:0	**Indirect IT-MV Data Length**
		Format: U10 FormatDesc: In bytes

MFD_IT_OBJECT

		This field provides the length in bytes of the indirect data, which contains all the MVs for the current MB (in any partitioning and subpartitioning form). A value zero indicates that indirect data fetching is disabled – subsequently, the Indirect IT-MV Data Start Address field is ignored. This field must have the same alignment as the Indirect Object Data Start Address.AVC-IT Mode: It must be DWord aligned (since each MV is 4bytes in size)Driver has to derived this field from MVsize (MVquantity in DXVA, exact size) *4 bytes per MV.This field is only valid in AVC decoder IT mode (VC1 and MPEG uses inline MV data).
2	31:29	**Reserved** Project: All Format: MBZ
	28:0	**Indirect IT-MV Data Start Address Offset** This field specifies the memory starting address (offset) of the MV data to be fetched into the IT pipeline for processing. This pointer is relative to the Indirect IT-MV Object Base Address.Hardware ignores this field if indirect data is not present, i.e. the Indirect MV Data Length is set to 0. Alignment of this address depends on the mode of operation.AVC-IT Mode: It must be DWord aligned (since each MV is 4 bytes in size). This field is only valid in AVC decoder IT mode (VC1 and MPEG uses inline MV data). **Value** | **Name** [0,512MB) |
3	31:12	**Reserved** Project: All Format: MBZ
	11:0	**Indirect IT-COEFF Data Length** Project: All This field provides the length in bytes of the indirect data, which contains all the non-zero coefficients for the current MB. A value zero indicates that indirect data fetching is disabled – subsequently, the Indirect IT-COEFF Data Start Address field is ignored. Since each IT-COEFF data is 1 DW in size, with 12 bits, this field can be extended to support up to 4:4:4 format.(256 pixel * 3 byte pixel components * 4 bytes per coeff).This field must be integer multiple of 16-bytes for AVC (since each coefficient is 4 bytes in size).This field is only valid in AVC, VC1, MPEG2 decoder IT mode. **Value** | **Name** [0,3072] | In bytes [0, 256*3*4]
4	31:29	**Reserved** Project: All Format: MBZ
	28:0	**Indirect IT-COEFF Data Start Address Offset** Project: All This field specifies the memory starting address (offset) of the coeff data to be loaded into the IT pipeline for processing. This pointer is relative to the Indirect IT-COEFF Object Base Address.Hardware ignores this field if indirect IT-COEFF data is not present, i.e. the Indirect IT-COEFF Data Length is set to 0.This field must be DW aligned, since each coeff icient is 4 bytes in size.Driver will determine the Num of EOB 4x4/8x8 must match the block cbp flags, if not match, hardware cannot hang – add error handling.This field is only valid in AVC, VC1, MPEG2 decoder IT mode. **Value** | **Name** [0,512MB) |
5	31:6	**Reserved** Project: All Format: MBZ

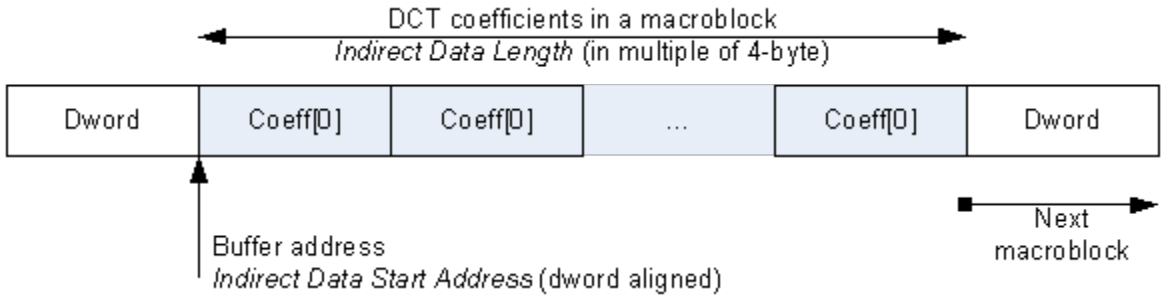
MFD_IT_OBJECT		
	5:0	**Indirect IT-DBLK Control Data Length** Project: All Format: U6 This field provides the length in bytes of the indirect data, which contains all the deblocker control information for the current MB (in 4x4 sub-block partitioning). A value zero indicates that indirect data fetching is disabled – subsequently, the Indirect IT-DBLK Data Start Address field is ignored. This field must have the same alignment as the Indirect IT-DBLK Data Start Address. It must be DWord aligned. Each Deblock Control Data record is 48 bytes or 12 DWords in size.This field is only valid in AVC decoder IT mode.
6	31:29	**Reserved** Format: MBZ
	28:0	**Indirect IT-DBLK Control Data Start Address Offset** Format: IndirectObjectBaseAddress[28:0] This field specifies the memory starting address (offset) of the Deblocker control data to be fetched into the IT Pipeline for processing. This pointer is relative to the Indirect IT-DBLK Object Base Address. Hardware ignores this field if indirect data is not present, ie. The indirect IT-DBLK Control Data Length is set to 0. It must be DWord aligned. Each Deblock Control Data record is 48 bytes or 12 DWords in size. This field is only valid in AVC decoder IT mode. **Value**: [0,512MB) — **Name**:
7..n	31:0	**Inline Data** Union for all 3 codecs Includes IT, MC, IntraPred inline data as well as Deblocker control information AVC-IT Modes: Hardware interprets this data in the specified format. VC1-IT Modes: Hardware interprets this data in the specified format. MV inline MPEG2-IT Modes: Hardware interprets this data in the specified format. (IS mode) MV inline For AVC there 7 DWords of inline data, hence N is equal to 13.

5.1.1.1 Common Indirect IT-COEFF Data Structure

Transform-domain residual data block in AVC-IT, VC1-IT and MPEG2-IT mode follows the same data structure.

The indirect IT-COEFF data start address in MFD_IT_OBJECT command specifies the doubleword aligned address of the first non-zero DCT coefficient of the first block of the macroblock. Only the non-zero coefficients are present in the data buffer and they are packed in the 8x8 block sequence of Y0, Y1, Y2, Y3, Cb4 and Cr5, as shown in *Common Indirect IT COEFF Data Structure*. When an 8x8 block is further subdivided into 4x4 subblocks, the coefficients, if present, are organized in the subblock order. The smallest subblock division is referred to as a **transform block**. The indirect IT-COEFF data length in the command includes all the non-zero coefficients for the macroblock. It must be doubleword aligned.

Structure of the IDCT Compressed Data Buffer



Each non-zero coefficient in the indirect data buffer is contained in a doubleword-size data structure consisting of the coefficient index, end of block (EOB) flag and the fixed-point coefficient value in 2's compliment form. As shown in *Common Indirect IT COEFF Data Structure*, *index* is the row major 'raster' index of the coefficient **within a transform block** (*please note that it is not converted to 8x8 block basis*). A coefficient is a 16-bit value in 2's complement.

Structure of a transform-domain residue unit

DWord	Bit	Description
0	31:16	**Transform-Domain Residual (coefficient) Value.** This field contains the value of the non-zero transform-domain residual data in 2's compliment.
	15:7	Reserved: MBZ
	6:1	**Index.** This field specifies the raster-scan address (raw address) of the coefficient within the transform block. For a coefficient at Cartesian location (row, column) = (y, x) in a transform block of width W, Index is equal to (y * W + x). For example, coefficient at location (row, column) = (0, 0) in a 4x4 transform block has an index of 0; that at (2, 3) has an index of 2*4 + 3 = 11. The valid range of this field depends on the size of the transform block. Format = U6 Range = [0, 63]
	0	**EOB (End of Block).** This field indicates whether the transform-domain residue is the last one of the current transform block.

Allowed transform block dimensions per coding standard

Transform Block Dimension	AVC	VC1	MPEG2
8x8	Yes	Yes	Yes
8x4	No	Yes	No
4x8	No	Yes	No
4x4	Yes	Yes	No

For AVC, there is intra16x16 mode, in which the DC Luma coefficients of all 4x4 sub-blocks within the current MB are sent separately in its own 4x4 Luma block. As such, only 15 coefficients remains in each of the 16 4x4 Luma blocks.

5.1.1.2 Inline Data Description in AVC-IT Mode

The Inline Data includes all the required MB decoding states, extracted primarily from the Slice Data, MB Header and their derivatives. It provides information for the following operations:

1. Inverse Quantization
2. Inverse Transform
3. Intra and inter-Prediction decoding operations
4. Internal error handling

IT Mode supports only packed MV data as specified in the DXVA spec.

These state/parameter values may subject to change on a per-MB basis, and must be provided in each MFD_IT_OBJECT command. The values set for these variables are retained internally, until they are reset by hardware Asynchronous Reset or changed by the next MFC_AVC_PAK_OBJECT command.

The inline data has been designed to match the DXVA 2.0, with the exception of the starting byte (DW0:0-7) and the ending dword (DW7:0-31).

The Deblocker Filter Control flags (FilterInternalEdgesFlag, FilterTopMbEdgeFlag and FilterLeftMbEdgesFlag) are generated by H/W, which are depending on MbaffFrameFlag, CurrMbAddr, PicWidthInMbs and disable_deblocking_filter_idc states.

Current MB [x,y] address is not sent, it is assumed that the H/W will keep track of the MB count and current MB position internally.

DWord	Bit	Description
0	31:24	**MvQuantity** Specify the number of MVs (in unit of motion vector, 4 bytes each) to be fetched for motion compensation operation. Decoder IT mode only supports packed MV format (DXVA). This field specifies the exact number of MVs present for the current MB. For a P-Skip MB, there is still 1 MV being sent (Skip MV is sent explicitly); for a B-Direct/Skip MB, there are 2 MVs being sent. For an Intra-MB, MvQuantity is set to 0. MvQuantity = 0, signifies there is no MV indirect data for the current MB. This field must be set in consistent with **Indirect MV Data Length**, so as not to exceed its bound Unsigned.
	23:20	Reserved MBZ (DXVA)
	19	**DcBlockCodedYFlag** 1 – the 4x4 DC-only Luma sub-block of the Intra16x16 coded MB is present; it is still possible that all DC coefficients are zero. 0 – no 4x4 DC-only Luma sub-block is present; either not in Intra16x16 MB mode or all DC coefficients are zero.
	18	**DcBlockCodedCbFlag** For 4:2:0 case :

DWord	Bit	Description
		1 – the 2x2 DC-only Chroma Cb sub-block of all coded MB (any type) is present; it is still possible that all DC coefficients are zero. 0 – no 2x2 DC-only Chroma Cb sub-block is present; all DC coefficients are zero.
	17	**DcBlockCodedCrFlag** For 4:2:0 case : 1 – the 2x2 DC-only Chroma Cr sub-block of all coded MB (any type) is present; it is still possible that all DC coefficients are zero. 0 – no 2x2 DC-only Chroma Cr sub-block is present; all DC coefficients are zero.
	16	Reserved MBZ (DXVA)
	15	**Transform8x8Flag** 0: indicates the current MB is coded with 4x4 transform and therefore the luma residuals are presented in 4x4 blocks. 1: indicates the current MB is coded with 8x8 transform and therefore the luma residuals are presented in 8x8 blocks. Same as the transform_szie_8x8_flag syntax element in AVC spec.
	14	**MbFieldFlag** This field specifies whether current macroblock is coded as a field or frame macroblock in MBAFF mode. 1 = Field macroblock 0 = Frame macroblock This field is exactly the same as FIELD_PIC_FLAG syntax element in non-MBAFF mode. Same as the mb_field_decoding_flag syntax element in AVC spec.
	13	**IntraMbFlag** This field specifies whether the current macroblock is an Intra (I) macroblock. 0 – not an intra MB 1 – is an intra MB I_PCM is considered as Intra MB. For I-picture MB (IntraPicFlag =1), this field must set to 1. This flag must be set in consistent with the

DWord	Bit	Description
		interpretation of MbType (inter or intra modes).
	12:8	**MbType** This field carries the Macroblock Type. The meaning depends on IntraMbFlag. If IntraMbFlag is 1, this field is the intra macroblock type as defined in *Macroblock Type for Intra Cases*. If IntraMbFlag is 0, this field is the inter macroblock type as defined in the first two columns of *Macroblock Type for Inter Cases*. All macroblock types in a P Slice are mapped into the corresponding types in a B Slice. Skip and Direct modes are converted into its corresponding processing modes. Programming note: It is exactly matched with that of DXVA 2.0.
	7	**FieldMbPolarityFlag** This field indicates the field polarity of the current macroblock. Within a MbAff frame picture, this field may be different per macroblock and is set to 1 only for the second macroblock in a MbAff pair if FieldMbFlag is set. Otherwise, it is set to 0. Within a field picture, this field is set to 1 if the current picture is the bottom field picture. Otherwise, it is set to 0. It is a constant for the whole field picture. This field is only valid for MBAFF frame picture. It is reserved and set to 0 for a progressive frame picture or a field picture. 0 = Current macroblock is a field macroblock from the **top** field (first in a MBAFF pair) 1 = Current macroblock is a field macroblock from the **bottom** field (second in a MBAFF pair)
	6	**IsLastMB** 1 – the current MB is the last MB in the current Slice 0 – the current MB is not the last MB in the current Slice
	5-4	Reserved MBZ (Intel encoder)
	3:0	Reserved MBZ (DXVA Decoder)
1	31:16	**CbpY[bit 15:0] (Coded Block Pattern Y)** For 4x4 sub-block (when **Transform8x8flag** =

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td></td><td></td><td>0 or in intra16x16) :

16-bit cbp, one bit for each 4x4 Luma sub-block (not including the DC 4x4 Luma block in intra16x16) in a MB. The 4x4 Luma sub-blocks are numbered as

<table>
<tr><td>blk0</td><td>1</td><td>4</td><td>5</td><td>bit15</td><td>14</td></tr>
<tr><td>blk2</td><td>3</td><td>6</td><td>7</td><td>bit13</td><td>12</td></tr>
<tr><td>blk8</td><td>9</td><td>12</td><td>13</td><td>bit7</td><td>6</td></tr>
<tr><td>blk10</td><td>11</td><td>14</td><td>15</td><td>bit 5</td><td>4</td></tr>
</table>

The cbpY bit assignment is cbpY bit [15 – X] for sub-block_num X.

For 8x8 block (when Transform8x8flag = 1)

Only the lower 4 bits [3:0] are valid; the remaining upper bits [15:4] are ignored. The 8x8 Luma blocks are numbered as

<table>
<tr><td>blk0</td><td>1</td><td>bit3</td></tr>
<tr><td>blk2</td><td>3</td><td>bit1</td></tr>
</table>

The cbpY bit assignment is cbpY bit [3 – X] for block_num X.

0 in a bit – indicates the corresponding 8x8 block or 4x4 sub-block is not present (because all coefficient values are zero)

1 in a bit – indicates the corresponding 8x8 block or 4x4 sub-block is present (although it is still possible to have all its coefficients be zero – bad coding).</td></tr>
<tr><td></td><td>15:8</td><td>VertOrigin (Vertical Origin). This field specifies the vertical origin of current macroblock in the destination picture in units of macroblocks.

For field macroblock pair in MBAFF frame, the vertical origins for both macroblocks should be set as if they were located in corresponding field pictures. For example, for field macroblock pair originated at (16, 64) pixel location in an MBAFF frame picture, the Vertical Origin for both macroblocks should be set as 2 (macroblocks). Whether the current macroblock is the first/second (top/bottom) in a MBAFF pair is specified by FieldMbPolarityFlag.

The macroblocks with (VertOrigin, HorzOrigin) must be delivered in the strict order as coded in the bitstream (raster order for progressive frame or field pictures and MBAFF pair order for MBAFF pictures). No gap is allowed. Otherwise, hardware behavior</td></tr>
</table>

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td></td><td></td><td>is undefined.

Format = U8 in unit of macroblock.</td></tr>
<tr><td></td><td>7:0</td><td>HorzOrigin (Horizontal Origin). This field specifies the horizontal origin of current macroblock in the destination picture in units of macroblocks.

Format = U8 in unit of macroblock.</td></tr>
<tr><td>2</td><td>31:16</td><td>CbpCr (Coded Block Pattern Cr 4:2:0-only)

Only the lower 4 bits [3:0] are valid; the remaining upper bits [15:4] are ignored (only valid for 4:2:2 and 4:4:4). The 4x4 Chroma Cr sub-blocks are numbered as
<table><tr><td>blk0</td><td>1</td><td>bit3</td></tr><tr><td>blk2</td><td>3</td><td>bit1</td></tr></table>The cbpCr bit assignment is cbpCr bit [3 – X] for sub-block_num X.

0 in a bit – indicates the corresponding 4x4 sub-block is not present (because all coefficient values are zero)

1 in a bit – indicates the corresponding 4x4 sub-block is present (although it is still possible to have all its coefficients be zero – bad coding).

For monochrome, this field is ignored.</td></tr>
<tr><td></td><td>15-0</td><td>CbpCb (Coded Block Pattern Cb 4:2:0-only)

Only the lower 4 bits [3:0] are valid; the remaining upper bits [15:4] are ignored (only valid for 4:2:2 and 4:4:4). The 4x4 Chroma Cb sub-blocks are numbered as
<table><tr><td>blk0</td><td>1</td><td>bit3</td></tr><tr><td>blk2</td><td>3</td><td>bit1</td></tr></table>The cbpCb bit assignment is cbpCb bit [3 – X] for sub-block_num X.

0 in a bit – indicates the corresponding 4x4 sub-block is not present (because all coefficient values are zero)

1 in a bit – indicates the corresponding 4x4 sub-block is present (although it is still possible to have all its coefficients be zero – bad coding).

For monochrome, this field is ignored.</td></tr>
</table>

DWord	Bit	Description
3	31:24	**Reserved** MBz
	23:16	**QpPrimeCr** Driver is responsible for deriving the QpPrimeCr from QpPrimeY. For 8-bit pixel data, QpCr is the same as QpPrimeCr, and it takes on a value in the range of 0 to 51, positive integer.
	15:8	**QpPrimeCb** Driver is responsible for deriving the QpPrimeCb from QpPrimeY. For 8-bit pixel data, QpCb is the same as QpPrimeCb, and it takes on a value in the range of 0 to 51, positive integer.
	7:0	**QpPrimeY** This is the per-MB QP value specified for the current MB. For 8-bit pixel data, QpY is the same as QpPrimeY, and it takes on a value in the range of 0 to 51, positive integer.
4 to 6	31:0 Each	For intra macroblocks, definition of these fields are specified in Inline data subfields for an Intra Macroblock For inter macroblocks, definition of these fields are specified in Inline data subfields for an Inter Macroblock

5.1.1.3 Indirect Data Format in AVC-IT Mode

Indirect data in AVC-IT mode consist of Motion Vectors, Transform-domain Residue (Coefficient) and ILDB control data. All three data records have variable size. Size of each Motion Vector record is determined by the MvQuantity value as shown in *Indirect Data Format in AVC IT Mode*. ILDB control record is fixed at the same size for all MBs in a picture. Coefficient data record is variable size per MB, since it may only consist of non-zero coefficients.

Each MV is represented in 4 bytes, in the form of

Lower 2 bytes : horizontal MVx component in q-pel units

Upper 2 bytes : vertical MVy component in q-pel units

Integer distance is measured in unit of samples in the frame or field grid position.

Chroma MVs are not sent and are derived in the H/W.

Indirect MV record size in AVC-IT mode

Macroblock Type	MVQuant
BP_L0_16x16	1
B_L1_16x16	1
B_Bi_16x16	2
BP_L0_L0_16x8	2
BP_L0_L0_8x16	2
B_L1_L1_16x8	2
B_L1_L1_8x16	2
B_L0_L1_16x8	2
B_L0_L1_8x16	2
B_L1_L0_16x8	2
B_L1_L0_8x16	2
B_L0_Bi_16x8	3
B_L0_Bi_8x16	3
B_L1_Bi_16x8	3
B_L1_Bi_8x16	3
B_Bi_L0_16x8	3
B_Bi_L0_8x16	3
B_Bi_L1_16x8	3
B_Bi_L1_8x16	3
B_Bi_Bi_16x8	4
B_Bi_Bi_8x16	4
BP_8x8	Sum

For macroblock type of BP_8x8, MvQuant takes the sum of value MvQ[i] of the four individual 8x8 sub macroblocks.

SubMbShape[i]	SubMbPredMode[i]	Description	MvQ[i]
0	0	BP_L0_8x8	1
0	1	B_L1_8x8	1
0	2	B_BI_8x8	2
1	0	BP_L0_8x4	2
1	1	B_L1_8x4	2
1	2	B_BI_8x4	4
2	0	BP_L0_4x8	2
2	1	B_L1_4x8	2
2	2	B_BI_4x8	4
3	0	BP_L0_4x4	4
3	1	B_L1_4x4	4
3	2	B_BI_4x4	8

Indirect data Deblocking Filter Control block in AVC-IT mode:

AVC Deblocker Control Data record has a fixed size for each MB in a picture and is 48 bytes or 12 Dwords in size.

DWord	Bit	Description
0	31:24	Reserved : MBZ (DXVA Decoder)
	23	**FilterTopMbEdgeFlag**
	22	**FilterLeftMbEdgeFlag**
	21	**FilterInternal4x4EdgesFlag**
	20	**FilterInternal8x8EdgesFlag**
	19	**FieldModeAboveMbFlag**
	18	**FieldModeLeftMbFlag**
	17	**FieldModeCurrentMbFlag**
	16	**MbaffFrameFlag** (DXVA Decoder reserved bit)
	15:8	**VertOrigin** Current MB y position (address)
	7:0	**HorzOrigin** Current MB x position (address)
1	31:30	**bS_h13** 2-bit boundary strength for internal top horiz 4-pixel edge 3
	29:28	**bS_h12** 2-bit boundary strength for internal top horiz 4-pixel edge 2
	27:26	**bS_h11** 2-bit boundary strength for internal top horiz 4-pixel edge 1
	25:24	**bS_h10** 2-bit boundary strength for internal top horiz 4-pixel edge 0
	23:22	**bS_v33** 2-bit boundary strength for internal right vert 4-pixel edge 3

DWord	Bit	Description
	21:20	**bS_v23** 2-bit boundary strength for internal right vert 4-pixel edge 2
	19:18	**bS_v13** 2-bit boundary strength for internal right vert 4-pixel edge 1
	17:16	**bS_v03** 2-bit boundary strength for internal right vert 4-pixel edge 0
	15:14	**bS_v32** 2-bit boundary strength for internal mid vert 4-pixel edge 3
	13:12	**bS_v22** 2-bit boundary strength for internal mid vert 4-pixel edge 2
	11:10	**bS_v12** 2-bit boundary strength for internal mid vert 4-pixel edge 1
	9:8	**bS_v02** 2-bit boundary strength for internal mid vert 4-pixel edge 0
	7:6	**bS_v31** 2-bit boundary strength for internal left vert 4-pixel edge 3
	5:4	**bS_v21** 2-bit boundary strength for internal left vert 4-pixel edge 2
	3:2	**bS_v11** 2-bit boundary strength for internal left vert 4-pixel edge 1
	1:0	**bS_v01** 2-bit boundary strength for internal left vert 4-pixel edge 0
2	31:28	**bS_v30_0** 4-bit boundary strength for Left0 4-pixel edge 3 (MSbit is wasted)
	17:24	**bS_v20_0** 4-bit boundary strength for Left0 4-pixel edge 2 (MSbit is wasted)
	23:20	**bS_v10_0** 4-bit boundary strength for Left0 4-pixel edge 1 (MSbit is wasted)
	19:16	**bS_v00_0** 4-bit boundary strength for Left0 4-pixel edge 0 (MSbit is wasted)
	15:14	**bS_h33** 2-bit boundary strength for internal bot horiz 4-pixel edge 3
	13:12	**bS_h32** 2-bit boundary strength for internal bot horiz 4-pixel edge 2
	11:10	**bS_h31** 2-bit boundary strength for internal bot horiz 4-pixel edge 1
	9:8	**bS_h30** 2-bit boundary strength for internal bot horiz 4-pixel edge 0
	7:6	**bS_h23** 2-bit boundary strength for internal mid horiz 4-pixel edge 3
	5:4	**bS_h22** 2-bit boundary strength for internal mid horiz 4-pixel edge 2
	3:2	**bS_h21** 2-bit boundary strength for internal mid horiz 4-pixel edge 1
	1:0	**bS_h20** 2-bit boundary strength for internal mid horiz 4-pixel edge 0
3	31:28	**bS_h03_0** 4-bit boundary strength for Top0 4-pixel edge 3 (MSbit is wasted)
	27:24	**bS_h02_0** 4-bit boundary strength for Top0 4-pixel edge 2 (MSbit is wasted)
	23:20	**bS_h01_0** 4-bit boundary strength for Top0 4-pixel edge 1 (MSbit is wasted)

DWord	Bit	Description
	19:16	**bS_h00_0** 4-bit boundary strength for Top0 4-pixel edge 0 (MSbit is wasted)
	15:12	**bS_v03** 4-bit boundary strength for Left1 4-pixel edge 3 (MSbit is wasted)
	11:8	**bS_v02** 4-bit boundary strength for Left1 4-pixel edge 2 (MSbit is wasted)
	7:4	**bS_v01** 4-bit boundary strength for Left1 4-pixel edge 1 (MSbit is wasted)
	3:0	**bS_v00** 4-bit boundary strength for Left1 4-pixel edge 0 (MSbit is wasted)
4	31:24	**bIndexBinternal_Y** Internal index B for Y
	23:16	**bIndexAinternal_Y** Internal index A for Y
	15:12	**bS_h03_1** 4-bit boundary strength for Top1 4-pixel edge 3 (MSbit is wasted)
	11:8	**bS_h02_1** 4-bit boundary strength for Top1 4-pixel edge 2 (MSbit is wasted)
	7:4	**bS_h01_1** 4-bit boundary strength for Top1 4-pixel edge 1 (MSbit is wasted)
	3:0	**bS_h00_1** 4-bit boundary strength for Top1 4-pixel edge 0 (MSbit is wasted)
5	31:24	**bIndexBleft1_Y**
	23:16	**bIndexAleft1_Y**
	15:8	**bIndexBleft0_Y**
	7:0	**bIndexAleft0_Y**
6	31:24	**bIndexBtop1_Y**
	23:16	**bIndexAtop1_Y**
	15:8	**bIndexBtop0_Y**
	7:0	**bIndexAtop0_Y**
7	31:24	**bIndexBleft0_Cb**
	23:16	**bIndexAleft0_Cb**
	15:8	**bIndexBinternal_Cb**
	7:0	**bIndexAinternal_Cb**
8	31:24	**bIndexBtop0_Cb**
	23:16	**bIndexAtop0_Cb**
	15:8	**bIndexBleft1_Cb**

DWord	Bit	Description
	7:0	**bIndexAleft1_Cb**
9	31:24	**bIndexBinternal_Cr**
	23:16	**bIndexAinternal_Cr**
	15:8	**bIndexBtop1_Cb**
	7:0	**bIndexAtop1_Cb**
10	31:24	**bIndexBleft1_Cr**
	23:16	**bIndexAleft1_Cr**
	15:8	**bIndexBleft0_Cr**
	7:0	**bIndexAleft0_Cr**
11	31:24	**bIndexBtop1_Cr**
	23:16	**bIndexAtop1_Cr**
	15:8	**bIndexBtop0_Cr**
	7:0	**bIndexAtop0_Cr**

5.1.1.4 Inline Data Description in VC1-IT Mode

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>+0</td><td>31:28</td><td>MvFieldSelect. A bit-wise representation indicating which field in the reference frame is used as the reference field for current field. It’s only used in decoding interlaced pictures.

This field is valid for non-intra macroblock only.

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>28</td><td>Forward predict of current frame/field or TOP field of interlace frame, or block 0 in 4MV mode.</td></tr>
<tr><td>29</td><td>Backward predict of current frame/field or TOP field of interlace frame, or forward predict for block 1 in 4MV mode.</td></tr>
<tr><td>30</td><td>Forward predict of BOTTOM field of interlace frame, or block 2 in 4MV mode.</td></tr>
<tr><td>31</td><td>Backward predict of BOTTOM field of interlace frame, or forward predict for block 3 in 4MV mode.</td></tr>
</table>

Each corresponding bit has the following indication.

0 = The prediction is taken from the top reference field.

1 = The prediction is taken from the bottom reference field.</td></tr>
<tr><td></td><td>27</td><td>Reserved. MBZ</td></tr>
<tr><td></td><td>26</td><td>MvFieldSelectChroma . This field specifies the polarity of reference field for chroma blocks when their motion vector is derived in Motion4MV mode for interlaced (field) picture.</td></tr>
</table>

DWord	Bit	Description
		Non-intra macroblock only. This field is derived from MvFieldSelect. 0 = The prediction is taken from the top reference field. 1 = The prediction is taken from the bottom reference field.
	25:24	**MotionType – Motion Type** For frame picture, a macroblock may only be either 00 or 10. For interlace picture, a macroblock may be of any motion types. It can be 01 if and only if DctType is 1. This field is 00 if and only if IntraMacroblock is 1. 00 = Intra 01 = Field Motion. 10 = Frame Motion or no motion. Others = Reserved.
	23	**Reserved.** MBZ
	22	**MvSwitch.** This field specifies whether the prediction needs to be switched from forward to backward or vice versa for single directional prediction for top and bottom fields of interlace frame B macroblocks. 0 = No directional prediction switch from top field to bottom field 1 = Switch directional prediction from top field to bottom field
	21	**DctType.** This field specifies whether the residual data is coded as field residual or frame residual for interlaced picture. This field can be 1 only if MotionType is 00 (intra) or 01 (field motion). For progressive picture, this field must be set to '0', i.e. all macrobalcoks are frame macroblock. 0 = Frame residual type. 1 = Field residual type.
	20	**OverlapTransform.** This field indicates whether overlap smoothing filter should be performed on I-block boundaries. 0 = No overlap smoothing filter. 1 = Overlap smoothing filter performed.
	19	**Motion4MV.** This field indicates whether current macroblock a progressive P picture uses 4 motion vectors, one for each luminance block. It's only valid for progressive P-picture decoding. Otherwise, it is reserved and MBZ. For example, with MotionForward is 0, this field must also be set to 0. 0 = 1MV-mode. 1 = 4MV-mode.
	18	**MotionBackward.** This field specifies whether the backward motion vector is active for B-picture. This field must be 0 if Motion4MV is 1 (no backward motion vector in 4MV-mode). 0 = No backward motion vector. 1 = Use backward motion vector(s).

DWord	Bit	Description
	17	**MotionForward.** This field specifies whether the forward motion vector is active for P and B pictures. 0 = No forward motion vector. 1 = Use forward motion vector(s).
	16	**IntraMacroblock.** This field specifies if the current macroblock is intra-coded. When set, Coded Block Pattern is ignored and no prediction is performed (i.e., no motion vectors are used). For field motion, this field indicates whether the top field of the macroblock is coded as intra. 0 = Non-intra macroblock. 1 = Intra macroblock.
	15:12	**LumaIntra8x8Flag – Luma Intra 8x8 Flag** This field specifies whether each of the four 8x8 luminance blocks are intra or inter coded when Motion4MV is set to 4MV-Mode. Each bit corresponds to one block. “0” indicates the block is inter coded and ‘1’ indicates the block is intra coded. When Motion4MV is not 4MV-Mode, this field is reserved and MBZ. Bit 15: Y0 Bit 14: Y1 Bit 13: Y2 Bit 12: Y3
	11:6	**CBP - Coded Block Pattern** This field specifies whether the 8x8 residue blocks in the macroblock are present or not. Each bit corresponds to one block. “0” indicates residue block isn’t present, “1” indicates residue block is present. Note: For each block in an intra-coded macroblock or an intra-coded block in a P macroblock in 4MV-Mode, the corresponding CBP must be 1. Subsequently, there must be at least one coefficient (this coefficient might be zero) in the indirect data buffer associated with the bock (i.e. residue block must be present). Bit 11: Y0 Bit 10: Y1 Bit 9: Y2 Bit 8: Y3 Bit 7: Cb4 Bit 6: Cr5
	5	**ChromaIntraFlag - Derived Chroma Intra Flag** This field specifies whether the chroma blocks should be treated as intra blocks based on motion vector derivation process in 4MV mode. 0 = Chroma blocks are not coded as intra. 1 = Chroma blocks are coded as intra

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td></td><td>4</td><td>LastRowFlag – Last Row Flag
This field indicates that the current macroblock belongs to the last row of the picture.
This field may be used by the kernel to manage pixel output when overlap transform is on.
0 = Not in the last row
1 = In the last row</td></tr>
<tr><td></td><td>3</td><td>LastMBInRow – This field indicates the last MB in row flag.</td></tr>
<tr><td></td><td>2:0</td><td>Reserved. MBZ</td></tr>
<tr><td>+1</td><td>32:26</td><td>Reserved. MBZ</td></tr>
<tr><td></td><td>15:8</td><td>VertOrigin - Vertical Origin
In unit of macroblocks relative to the current picture (frame or field).</td></tr>
<tr><td></td><td>7:0</td><td>HorzOrigin - Horizontal Origin
In unit of macroblocks.</td></tr>
<tr><td>+2</td><td>31:16</td><td>MotionVector[0].Vert</td></tr>
<tr><td></td><td>15:0</td><td>MotionVector[0].Horz</td></tr>
<tr><td>+3</td><td>31:0</td><td>MotionVector[1]</td></tr>
<tr><td>+4</td><td>31:0</td><td>MotionVector[2]</td></tr>
<tr><td>+5</td><td>31:0</td><td>MotionVector[3]</td></tr>
<tr><td>+6</td><td>31:0</td><td>MotionVectorChroma
This field is not valid for a field motion in an interlaced frame picture where 4 MVs for chroma blocks.
Notes: This field is derived from MotionVector[3:0] as described in the following section.</td></tr>
<tr><td>+7</td><td>31:24</td><td>Subblock Code for Y3
The following subblock coding definition applies to all 6 subblock coding bytes. Bits 7:6 are reserved.
<table>
<tr><th colspan="2">Subblock Partitioning
(Bits [1:0])
Specify Transform uses for an 8x8 block</th><th colspan="4">Subblock Present
(0 means not present, 1 means present)</th></tr>
<tr><th>Bits [1:0]</th><th>Meaning</th><th>Bit 2</th><th>Bit 3</th><th>Bit 4</th><th>Bit 5</th></tr>
<tr><td>00</td><td>Single 8x8 block (sb0)</td><td>Sb0</td><td>Don’t care</td><td>Don’t care</td><td>Don’t care</td></tr>
<tr><td>01</td><td>Two 8x4 subblocks (sb0-1)</td><td>Sb1 (bot)</td><td>Sb0 (top)</td><td>Don’t care</td><td>Don’t care</td></tr>
<tr><td>10</td><td>Two 4x8 subblocks (sb0-1)</td><td>Sb1 (right)</td><td>Sb0 (left)</td><td>Don’t care</td><td>Don’t care</td></tr>
<tr><td>11</td><td>Four 4x4 subblocks (sb0-3)</td><td>Sb3 (lower</td><td>Sb2 (lower</td><td>Sb1 (upper</td><td>Sb0 (upper</td></tr>
</table></td></tr>
</table>

DWord	Bit	Description
		right) left) right) left)
	23:16	**Subblock Code for Y2**
	15:8	**Subblock Code for Y1**
	7:0	**Subblock Code for Y0**
+8	31:16	**Reserved.** MBZ
	15:8	**Subblock Code for Cr**
	7:0	**Subblock Code for Cb**
+9	31:24	**ILDB control data for block Y3**
	23:16	**ILDB control data for block Y2**
	15:8	**ILDB control data for block Y1**
	7:0	**ILDB control data for block Y0**
+10	31:16	**Reserved**
	15:8	**ILDB control data for Cr block**
	7:0	**ILDB control data for Cb block**

5.1.1.5 Indirect Data Format in VC1-IT Mode

VC1-IT mode only contains IT-COEFF indirect data which is described in *Common Indirect IT COEFF Data Structure*.

5.1.1.6 Inline Data Description in MPEG2-IT Mode

The content in this command is similar to that in the MEDIA_OBJECT command in IS mode described in the Media Chapter.

Each MFD_IT_OBJECT command corresponds to the processing of one macroblock. Macroblock parameters are passed in as inline data and the non-zero DCT coefficient data for the macroblock is passed in as indirect data.

Inline Data Description in MPEG2 IT Mode depicts the inline data format. Inline data starts at dword 7 of MFD_IT_OBJECT command. There are 7 dwords total.

Inline data in MPEG2-IT Mode

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>+0</td><td>31:28</td><td>Motion Vertical Field Select. A bit-wise representation of a long [2][2] array as defined in §6.3.17.2 of the ISO/IEC 13818-2 (see also §7.6.4).
<table>
<tr><th>Bit</th><th>MVector[r]</th><th>MVector[s]</th><th>MotionVerticalFieldSelect Index</th></tr>
<tr><td>28</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>29</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>30</td><td>1</td><td>0</td><td>2</td></tr>
<tr><td>31</td><td>1</td><td>1</td><td>3</td></tr>
</table>
Format = MC_MotionVerticalFieldSelect.
0 = The prediction is taken from the top reference field.
1 = The prediction is taken from the bottom reference field.</td></tr>
<tr><td></td><td>27</td><td>Reserved (was Second Field)</td></tr>
<tr><td></td><td>26</td><td>Reserved. (HWMC mode)</td></tr>
<tr><td></td><td>25:24</td><td>Motion Type. When combined with the destination picture type (field or frame) this Motion Type field indicates the type of motion to be applied to the macroblock. See ISO/IEC 13818-2 §6.3.17.1, Tables 6-17, 6-18. In particular, the device supports dual-prime motion prediction (11) in both frame and field picture type.
Format = MC_MotionType
<table>
<tr><th>Value</th><th>Destination = Frame
Picture_Structure = 11</th><th>Destination = Field
Picture_Structure != 11</th></tr>
<tr><td>‘00’</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>‘01’</td><td>Field</td><td>Field</td></tr>
<tr><td>‘10’</td><td>Frame</td><td>16x8</td></tr>
<tr><td>‘11’</td><td>Dual-Prime</td><td>Dual-Prime</td></tr>
</table></td></tr>
<tr><td></td><td>23:22</td><td>Reserved. (Scan method)</td></tr>
<tr><td></td><td>21</td><td>DCT Type. This field specifies the DCT type of the current macroblock. The kernel should ignore this field when processing Cb/Cr data. See ISO/IEC 13818-2 §6.3.17.1. This field is zero if Coded Block Pattern is also zero (no coded blocks present).
0 = MC_FRAME_DCT (Macroblock is frame DCT coded).
1 = MC_FIELD_DCT (Macroblock is field DCT coded).</td></tr>
<tr><td></td><td>20</td><td>Reserved (was Overlap Transform - H261 Loop Filter).</td></tr>
<tr><td></td><td>19</td><td>Reserved (was 4MV Mode - H263/)</td></tr>
<tr><td></td><td>18</td><td>Macroblock Motion Backward. This field specifies if the backward motion vector is active. See ISO/IEC 13818-2 Tables B-2 through B-4.
0 = No backward motion vector.
1 = Use backward motion vector(s).</td></tr>
<tr><td></td><td>17</td><td>Macroblock Motion Forward. This field specifies if the forward motion vector is active. See ISO/IEC 13818-2 Tables B-2 through B-4.
0 = No forward motion vector.</td></tr>
</table>

DWord	Bit	Description
		1 = Use forward motion vector(s).
	16	**Macroblock Intra Type.** This field specifies if the current macroblock is intra-coded. When set, Coded Block Pattern is ignored and no prediction is performed (i.e., no motion vectors are used). See *ISO/IEC 13818-2* Tables B-2 through B-4. 0 = Non-intra macroblock. 1 = Intra macroblock.
	15:12	**Reserved : MBZ**
	11:6	**Coded Block Pattern.** This field specifies whether blocks are present or not. Format = 6-bit mask. Bit 11: Y0 Bit 10: Y1 Bit 9: Y2 Bit 8: Y3 Bit 7: Cb4 Bit 6: Cr5
	5:4	Reserved. (Quantization Scale Code)
	3	LastMBInRow – This field indicates the last MB in each row.
	2:0	Reserved: MBZ
+1	31:16	**Reserved : MBZ**
	15:8	**VertOrigin - Vertical Origin** In unit of macroblocks relative to the current picture (frame or field).
	7:0	**HorzOrigin - Horizontal Origin** In unit of macroblocks.
+2	31:16	**Motion Vectors – Field 0, Forward, Vertical Component.** Each vector component is a 16-bit two's-complement value. The vector is relative to the current macroblock location. According to ISO/IEC 13818-2 Table 8, the valid range of each vector component is [-2048, +2047.5], implying a format of s11.1. However, it should be noted that motion vector values are sign extended to 16 bits.
	15:0	**Motion Vectors – Field 0, Forward, Horizontal Component**
+3	31:16	**Motion Vectors – Field 0, Backward, Vertical Component**
	15:0	**Motion Vectors – Field 0, Backward, Horizontal Component**
+4	31:16	**Motion Vectors – Field 1, Forward, Vertical Component**
	15:0	**Motion Vectors – Field 1, Forward, Horizontal Component**
+5	31:16	**Motion Vectors – Field 1, Backward, Vertical Component**

DWord	Bit	Description
	15:0	**Motion Vectors – Field 1, Backward, Horizontal Component**

5.1.1.7 Indirect Data Format in MPEG2-IT Mode

MPEG2-IT mode only contains IT-COEFF indirect data which is described in Section *Common Indirect IT COEFF Data Structure*.

5.2 Session Decoder StreamOut Data Structure

When StreamOut is enabled, per MB intermediated decoded data (MVs, mb_type, MB qp, etc.) are sent to the memory in a fixed record format (and of fixed size). The per-MB records must be written in a strict raster order and with no gap (i.e. every MB regardless of its mb_type and slice type, must have an entry in the StreamOut buffer). Therefore, the consumer of the StreamOut data can offset into the StreamOut Buffer (**StreamOut Data Destination Base Address**) using individual MB addresses.

A StreamOut Data record format is detailed as follows:

DWord	Bit	Description
0	31:24	Format: U5, valid from 0 to 32
	23	**Reserved MBZ**
	22-20	**EdgeFilterFlag** (AVC) / **OverlapSmoothFilter** (VC1)
	19:17	**CodedPatternDC** (for AVC only, **111b** for others) The field indicates whether DC coefficients are sent.. 1 bit each for Y, U and V.
	16	**Reserved MBZ**
	15	**Transform8x8Flag** When it is set to 0, the current MB uses 4x4 transform. When it is set to 1, the current MB uses 8x8 transform. The transform_szie_8x8_flag syntax element, if present in the output bitstream, is the same as this field. However, whether transform_szie_8x8_flag is present or not in the output bitstream depends on several conditions: This field is only allowed to be set to 1 for two conditions: It must be 1 if **IntraMbFlag** = INTRA and **IntraMbMode** = INTRA_8x8 It may be 1 if **IntraMbFlag** = INTER and there is no sub partition size less than 8x8 Otherwise, this field must be set to 0. 0: 4x4 integer transform **1: 8x8 integer transform**
	14	**MbFieldFlag** This field specifies whether current macroblock is coded as a field or frame macroblock in MBAFF

DWord	Bit	Description
		mode. This field is exactly the same as FIELD_PIC_FLAG syntax element in non-MBAFF mode. Same as the mb_field_decoding_flag syntax element in AVC spec. 0 = Frame macroblock 1 = Field macroblock
	13	**IntraMbFlag** This field specifies whether the current macroblock is an Intra (I) macroblock. I_PCM is considered as Intra MB. For I-picture MB (IntraPicFlag =1), this field must be set to 1. This flag must be set in consistent with the interpretation of MbType (inter or intra modes). 0: INTER (inter macroblock) **1: INTRA (intra macroblock)**
	12:8	**MbType5Bits** This field is encoded to match with the best macroblock mode determined as described in the next section. It follows AVC encoding for inter and intra macroblocks.
	7	**MbPolarity** FieldMB Polarity **-** vctrl_vld_top_field AVC
	6	**Reserved MBZ**
	5:4	**IntraMbMode** This field is provided to carry information partially overlapped with MbType. This field is only valid if **IntraMbFlag** = INTRA, otherwise, it is ignored by hardware..
	3	**Reserved MBZ**
	2	**MbSkipFlag** Reserved MBZ (DXVA Encoder). HW (VDSunit) doesn't have skip MB info. It sets to 1 if any of the sub-blocks is inter, uses predicted MVs, and skips sending MVs explicitly in the code stream. Currently H/W can provide this flag and is defaulted to 0 always.
	1:0	**InterMbMode** This field is provided to carry redundant information as that in MbType. It also carries additional information such as skip. This field is only valid if **IntraMbFlag** =INTER, otherwise, it is ignored by hardware.
1	31:16	**MbYCnt (Vertical Origin).** This field specifies the vertical origin of current macroblock in the destination picture in units of macroblocks. Format = U8 in unit of macroblock.
	15:0	**MbXCnt (Horizontal Origin).** This field specifies the horizontal origin of current macroblock in the destination picture in units of macroblocks.

DWord	Bit	Description
		Format = U8 in unit of macroblock.
2	31	**Conceal MB Flag.** This field specifies if the current MB is a conceal MB, use in AVC/VC1/MPEG2 mode
	30	**Last MB of the Slice Flag.** This field indicate the current MB is a last MB of the slice. Use in AVC/VC1/MPEG2 mode.
	29:24	**Reserved**
	23:20	**CbpAcV** 0 in a bit – indicates the corresponding 8x8 block or 4x4 sub-block is not present (because all coefficient values are zero) 1 in a bit – indicates the corresponding 8x8 block or 4x4 sub-block is present (although it is still possible to have all its coefficients be zero – bad coding).
	19:16	**CbpAcU** 0 in a bit – indicates the corresponding 8x8 block or 4x4 sub-block is not present (because all coefficient values are zero) 1 in a bit – indicates the corresponding 8x8 block or 4x4 sub-block is present (although it is still possible to have all its coefficients be zero – bad coding).
	15:0	**CbpAcY** 0 in a bit – indicates the corresponding 8x8 block or 4x4 sub-block is not present (because all coefficient values are zero) 1 in a bit – indicates the corresponding 8x8 block or 4x4 sub-block is present (although it is still possible to have all its coefficients be zero – bad coding). Bit15=Y0Sub0, Bit0=Y3Sub3
3	31:28 AVC	**Skip8x8Pattern** (AVC) This field indicates whether each of the four 8x8 sub macroblocks is using the predicted MVs and will not be explicitly coded in the bitstream (the sub macroblock will be coded as direct mode). It contains four 1-bit subfields, corresponding to the 4 sub macroblocks in sequential order. The whole macroblock may be actually coded as B_Direct_16x16 or B_Skip, according to the macroblock type conversion rules described in a later sub section. This field is only valid for a B slice. It is ignored by hardware for a P slice. Hardware also ignores this field for an intra macroblock. 0 in a bit – Corresponding MVs are sent in the bitstream 1 in a bit – Corresponding MVs are not sent in the bitstream
	27:25	**Reserved**
	24:16	**NzCoefCountMB** – all coded coefficients input including AC/DC blocks in current MB. Range 0 to 384 (9 bits)
	15:8	**[IVB+] MbClock16** – MB compute clocks in 16-clock unit.

DWord	Bit	Description
	7	**mbz** (AVC) **/ QScaleType** (MPEG2)
	6:0	**QpPrimeY** (AVC) **/ QScaleCode** (MPEG2) The luma quantization index. This is the per-MB QP value specified for the current MB.
4 to 6	31:0 Each	For intra macroblocks, definition of these fields are specified in 1 For inter macroblocks, definition of these fields are specified in 2
7	31:24	**Reserved**
	23:20	**MvFieldSelect** (Ref polarity top or bottom bits) for VC1 and MPEG2 vcp_vds_mvdataR[162:159] VC1 vmd_vds_mt_vert_fld_selR[3:0] MPEG2
	19:12	**Reserved**
	11:10	**SubBlockCodeType V** (If 8x8, 8x4, 4x8, 4x4 type)
	9:8	**SubBlockCodeType U** (specifies 8x8, 8x4, 4x8, 4x4 type) VC1
	7:6	**SubBlockCodeType Y3** (specifies 8x8, 8x4, 4x8, 4x4 type) VC1
	5:4	**SubBlockCodeType Y2** (specifies 8x8, 8x4, 4x8, 4x4 type) VC1
	3:2	**SubBlockCodeType Y1** (specifies 8x8, 8x4, 4x8, 4x4 type) VC1
	1:0	**SubBlockCodeType Y0** (specifies 8x8, 8x4, 4x8, 4x4 type) VC1
Inter cases		
8	31:16	**MvFwd[0].y** – y-component of the forward motion vector of the 1st 8x8 or 1st 4x4 subblock
	15:0	**MvFwd[0].x** – x-component of the forward motion vector of the 1st 8x8 or 1st 4x4 subblock
9	31:0	**MvBck[0]** – the backward motion vector of the 1st 8x8 or 1st 4x4 subblock
10	31:0	**MvFwd[1]** – the forward motion vector of the 2nd 8x8 or 4th 4x4 subblock
11	31:0	**MvBck[1]** – the backward motion vector of the 2nd 8x8 or 4th 4x4 subblock
12	31:0	**MvFwd[2]** – the forward motion vector of the 3rd 8x8 or 8th 4x4 subblock
13	31:0	**MvBck[2]** – the backward motion vector of the 3rd 8x8 or 8th 4x4 subblock
14	31:0	**MvFwd[3]** – the forward motion vector of the 4th 8x8 or 12th 4x4 subblock
15	31:0	**MvBck[3]** – the backward motion vector of the 4th 8x8 or 12th 4x4 subblock
Intra Cases :		

DWord	Bit	Description
8 to 15	31:0	**Reserved MBZ**

The inline data content of Dwords 4 to 6 is defined either for intra prediction or for inter prediction, but not both.

Inline data subfields for an Intra Macroblock

4	31:16	**LumaIndraPredModes[1]** Specifies the Luma Intra Prediction mode for four 4x4 sub-block of a MB, 4-bit each. AVC : See the bit assignment table later in this section. VC1 : MBZ. MPEG2 : MBZ.
	15:0	**LumaIndraPredModes[0]** Specifies the Luma Intra Prediction mode for four 4x4 sub-block, four 8x8 block or one intra16x16 of a MB. 4-bit per 4x4 sub-block (Transform8x8Flag=0, Mbtype=0 and intraMbFlag=1) or 8x8 block (Transform8x8Flag=1, Mbtype=0, MbFlag=1), since there are 9 intra modes. 4-bit for intra16x16 MB (Transform8x8Flag=0, Mbtype=1 to 24 and intraMbFlag=1), but only the LSBit[1:0] is valid, since there are only 4 intra modes. AVC : See the bit assignment table later in this section. VC1 : MBZ. MPEG2 : MBZ.
5 AVC INTRA	31:16	**LumaIndraPredModes[3]** Specifies the Luma Intra Prediction mode for four 4x4 sub-block of a MB, 4-bit each. AVC : See the bit assignment table later in this section. VC1 : MBZ. MPEG2 : MBZ.
	15:0	**LumaIndraPredModes[2]** Specifies the Luma Intra Prediction mode for four 4x4 sub-block of a MB, 4-bit each. AVC : See the bit assignment later in this section. VC1 : MBZ. MPEG2 : MBZ.
6	31:8	Reserved (Reserved for encocder turbo mode **IntraResidueDataSize**, when this is not 0, optional residue data are provided to the PAK; Reserved for decoder)
	7:0	**MbIntraStruct** The IntraPredAvailFlags[4:0] have already included the effect of the constrained_intra_pred_flag. See the diagram later for the definition of neighbors position around the current MB or MB pair (in MBAFF mode).

		1 – IntraPredAvailFlagX, indicates the values of samples of neighbor X can be used in intra prediction for the current MB. 0 – IntraPredAvailFlagX, indicates the values of samples of neighbor X is not available for intra prediction of the current MB. IntraPredAvailFlag-A and -E can only be different from each other when constrained_intra_pred_flag is equal to 1 and mb_field_decoding_flag is equal to 1 and the value of the mb_field_decoding_flag for the macroblock pair to the left of the current macroblock is equal to 0 (which can only occur when MbaffFrameFlag is equal to 1). IntraPredAvailFlag-F is used only if o it is in MBAFF mode, i.e. **MbaffFrameFlag** = 1, o the current macroblock is of frame type**,** i.e. **MbFieldFag** = 0, and o the current macroblock type is Intra8x8, i.e. **IntraMbFlag** = INTRA, **IntraMbMode** = INTRA_8x8, and **Transform8x8Flag** = 1. In any other cases IntraPredAvailFlag-A shall be used instead.

Bits	IntraPredAvailFlags[4:0] Definition
7	**IntraPredAvailFlagF – F** (Left 8th row (-1,7) neighbor)
6	**IntraPredAvailFlagA – A** (Left neighbor top half)
5	**IntraPredAvailFlagE – E** (Left neighbor bottom half)
4	**IntraPredAvailFlagB – B** (Top neighbor)
3	**IntraPredAvailFlagC – C** (Top right neighbor)
2	**IntraPredAvailFlagD – D** (Top left corner neighbor)
1:0	**ChromaIntraPredMode –** 2 bits to specify 1 of 4 chroma intra prediction mode, see the table in later section.

Inline data subfields for an Inter Macroblock

4	31:24	Reserved: MBZ (DXVA Decoder)
	23:16	Reserved: MBZ (DXVA Decoder)
	15:8	**SubMbPredModes**[bit 7:0] (Sub Macroblock Prediction Mode) This field describes the prediction mode of the sub macroblocks (four 8x8 blocks). It contains four subfields each with 2-bits, corresponding to the 4 fixed size 8x8 sub macroblocks in sequential order. This field is provided for MB with sub_mb_type equal to BP_8x8 only (B_8x8 and P_8x8 as defined in DXVA) This field is derived from MbType for a non-BP_8x8 inter macroblock, and carries redundant information as MbType) Bits [1:0]: SubMbPredMode[0] – for 8x8 Block 0 Bits [3:2]: SubMbPredMode[1] – for 8x8 Block 1 Bits [5:4]: SubMbPredMode[2] – for 8x8 Block 2

		Bits [7:6]: SubMbPredMode[3] – for 8x8 Block 3 Blocks of the MB is numbered as follows : 0 1 2 3 Each 2-bit value [1:0] is defined as : 00 – Pred_L0 01 – Pred_L1 10 – BiPred **For VC1:** Bits [1:0]: “00”= Y0 Forward only, “01”= Y0 Backward only, “10”= Y0 Bi direction Bits [3:2]: SubMbPredMode[1] – for 8x8 Block 1 Bits [5:4]: SubMbPredMode[2] – for 8x8 Block 2 Bits [7:6]: SubMbPredMode[3] – for 8x8 Block 3
	7:0	**SubMbShape**[bit 7:0] (Sub Macroblock Shape) This field describes the sub-block partitioning of each sub macroblocks (four 8x8 blocks). It contains four subfields each with 2-bits, corresponding to the 4 fixed size 8x8 sub macroblocks in sequential order. This field is provided for MB with sub_mb_type equal to BP_8x8 only (B_8x8 and P_8x8 as defined in DXVA) This field is forced to 0 for a non-BP_8x8 inter macroblock, and effectively carries redundant information as MbType). ??? Bits [1:0]: SubMbShape[0] – for 8x8 Block 0 Bits [3:2]: SubMbShape[1] – for 8x8 Block 1 Bits [5:4]: SubMbShape[2] – for 8x8 Block 2 Bits [7:6]: SubMbShape[3] – for 8x8 Block 3 Blocks of the MB is numbered as follows : 0 1 2 3 Each 2-bit value [1:0] is defined as : 00 – SubMbPartWidth=8, SubMbPartHeight=8 01 – SubMbPartWidth=8, SubMbPartHeight=4 10 – SubMbPartWidth=4, SubMbPartHeight=8 11 – SubMbPartWidth=4, SubMbPartHeight=4 For VC-1, This field indicates the transformation types used for luma components, 2 bits for each 8x8.
5	31:24	**Frame Store ID L0[3]** Support up to 4 Frame store ID per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the frame Store ID into the Reference Picture List0 Table.

		Bit 7: **Must Be One**: (This is reserved for control fields in future extension, when reference index are generated instead of frame store ID) 1: indicate it is in Frame store ID format. 0: indicate it is in Reference Index format. Bit 6:5: reserved MBZ Bit 4:0 : Frame store index or Frame Store ID (Bit 4:1 is used to form the binding table index in intel implementation)
	23:16	**Frame Store ID L0[2]** Support up to 4 Frame store ID per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the frame Store ID into the Reference Picture List0 Table. Bit 7: **Must Be One**: (This is reserved for control fields in future extension, when reference index are generated instead of frame store ID) 1: indicate it is in Frame store ID format. 0: indicate it is in Reference Index format. Bit 6:5: reserved MBZ Bit 4:0 : Frame store index or Frame Store ID (Bit 4:1 is used to form the binding table index in intel implementation)
	15:8	**Frame Store ID L0[1]** Support up to 4 Frame store ID per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the frame Store ID into the Reference Picture List0 Table. Bit 7: **Must Be One**: (This is reserved for control fields in future extension, when reference index are generated instead of frame store ID) 1: indicate it is in Frame store ID format. 0: indicate it is in Reference Index format. Bit 6:5: reserved MBZ Bit 4:0 : Frame store index or Frame Store ID (Bit 4:1 is used to form the binding table index in intel implementation).
	7:0	**Frame Store ID L0[0]** Support up to 4 Frame store ID per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the frame Store ID into the Reference Picture List0 Table. Bit 7: **Must Be One**: (This is reserved for control fields in future extension, when reference index are generated instead of frame store ID) 1: indicate it is in Frame store ID format. 0: indicate it is in Reference Index format. Bit 6:5: reserved MBZ Bit 4:0 : Frame store index or Frame Store ID (Bit 4:1 is used to form the binding table index in intel implementation)

6	31:24	**Frame Store ID L1[3]** Support up to 4 Frame store ID per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the frame Store ID into the Reference Picture List0 Table. Bit 7: **Must Be One**: (This is reserved for control fields in future extension, when reference index are generated instead of frame store ID) 1: indicate it is in Frame store ID format. 0: indicate it is in Reference Index format. Bit 6:5: reserved MBZ Bit 4:0 : Frame store index or Frame Store ID (Bit 4:1 is used to form the binding table index in intel implementation)
	23:16	**Frame Store ID L1[2]** Support up to 4 Frame store ID per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the frame Store ID into the Reference Picture List0 Table. Bit 7: **Must Be One**: (This is reserved for control fields in future extension, when reference index are generated instead of frame store ID) 1: indicate it is in Frame store ID format. 0: indicate it is in Reference Index format. Bit 6:5: reserved MBZ Bit 4:0 : Frame store index or Frame Store ID (Bit 4:1 is used to form the binding table index in intel implementation)
	15:8	**Frame Store ID L1[1]** Support up to 4 Frame store ID per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the frame Store ID into the Reference Picture List0 Table. Bit 7: **Must Be One**: (This is reserved for control fields in future extension, when reference index are generated instead of frame store ID) 1: indicate it is in Frame store ID format. 0: indicate it is in Reference Index format. Bit 6:5: reserved MBZ Bit 4:0 : Frame store index or Frame Store ID (Bit 4:1 is used to form the binding table index in intel implementation)
	7:0	**Frame Store ID L1[0]** Support up to 4 Frame store ID per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the frame Store ID into the Reference Picture List0 Table. Bit 7: **Must Be One**: (This is reserved for control fields in future extension, when reference index are generated instead of frame store ID) 1: indicate it is in Frame store ID format. 0: indicate it is in Reference Index format.

		Bit 6:5: reserved MBZ Bit 4:0 : Frame store index or Frame Store ID (Bit 4:1 is used to form the binding table index in intel implementation)

5.3 Decoder Input Bitstream Formats

5.3.1 AVC Bitstream Formats – DXVA Short

Bitstream Buffer Address starts after the 3-byte start code, i.e. starts (and includes) at the NAL Header Byte. This byte must not be included in the Emulation Byte Detection Process.

5.3.2 AVC Bitstream Formats – DXVA Long

Bitstream Buffer Address starts after the 3-byte start code, i.e. starts (and includes) at the NAL Header Byte. This byte must not be included in the Emulation Byte Detection Process. Application will provide the Slice Header Skip Byte count (not including any possible Emulation Prevention Byte).

5.3.3 AVC Bitstream Formats – Intel Long

Obsolete – not supported.

5.3.4 VC1 Bitstream Formats – Intel Long

Bitstream starts right at the MB layer, with any emulation byte crossing the header and MB layer being removed by application and the data length is adjusted.

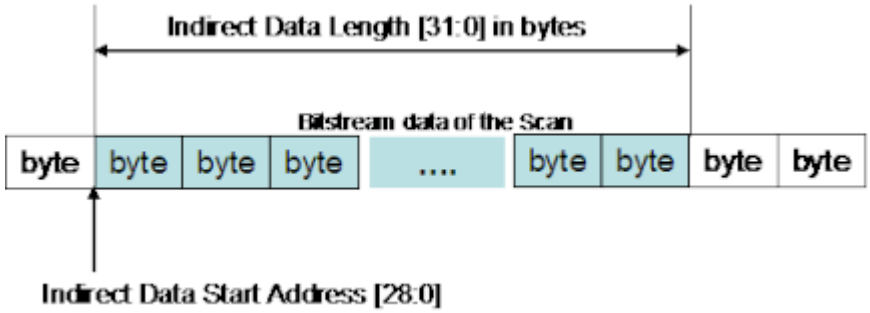
5.3.5 MPEG2 Bitstream Formats – DXVA1

Bitstream buffer starts right at the very first MB data.

5.3.6 JPEG Bitstream Formats – Intel

Bitstream buffer starts right at the very first MCU data of each Scan.

The indirect data start address in MFD_JPEG_BSD_OBJECT specifies the starting Graphics Memory address of the bitstream data that follows the Scan header. It provides the byte address for the first MCU of the Scan. Different from MFD_MPEG2_BSD_OBJECT command, First MCU Bit Offset does not need to be specified because it is always set to zero.



Indirect data buffer for a Scan

The indirect data length in MFD_JPEG_BSD_OBJECT provides the length in bytes of the bitstream data for the Scan excluding Scan header. It includes the first byte of the first macroblock and the last byte of the last macroblock in the Scan. The Figure illustrates these parameters for a slice data.

5.4 Concurrent, Multiple Video Stream Decoding Support

The natural place for switching across multiple streams is at the Slice boundary. Each Slice is a self-sustained unit of compressed video data and has no dependency with its neighbors (except for the Deblocking process). In addition, there is no interruptability within a Slice. However, when ILDB is invoked, the processing of some MBs will require neighbour MB information that crosses the Slice boundary. Hence, to limit the buffering requirement, in this version of hardware design, stream switching can only be performed at the picture boundary instead.

6. Encoder StreamOut Mode Data Structure Definition

When StreamOut is enabled, per MB (and/or per Slice, per Picture) intermediated coding data (e.g. bit allocated for each MB, etc.) are sent to the memory in a fixed record format (and of fixed size) from the PAK. The per-MB records must be written in a strict raster order and with no gap (i.e. every MB regardless of its mb_type and slice type, must have an entry in the StreamOut buffer). Therefore, the consumer of the StreamOut data can offset into the StreamOut Buffer (**StreamOut Data Destination Base Address**) using individual MB addresses.

Adding per macroblock stream out for PAK is for the following purposes:

- Immediate multi-pass PAK (without host or EU intervention)
    - 3200-bit conformance
    - Re-quantization
- Providing information for host for offline processing
- Providing information for updated QP's

The description for the fixed format PAK streamout record :

Streamout Pointer: Use the existing streamout pointer and enabler

Per Macroblock Information (a fixed size structure)

DWord	Bit	Description
0	31:24	**MbQpY** - Actual QPY used by the macroblock.
	23:16	**[IVB+] MbClock16** – MB compute clocks in 16-clock unit.
	15:8	Reserved : MBZ
	7:4	Reserved : **MBZ** (future conformance flags)
	3	**Reserved**
	2	**MbRcFlag:** MB level Rate control flag(pass through)The same value as **RateControlCounterEnable** of MFX_AVC_SLICE_STATE Command
	1	**MbInterConfFlag**: MB level InterMB conformance flag to trigger mutli-pass1- if total Bit Count of an inter macroblock is more than Inter Conformance Max size limit in the MFX_AVC_IMG_STATE Command
	0	**MbIntraConfFlag**: MB level IntraMB conformance flag to trigger mutli-pass1- if total Bit Count of an intra macroblock is more than Intra Conformance Max size limit in the MFX_AVC_IMG_STATE Command
1	31:29	**Reserved**
	28:16	**MbBits** : Total Bit Count for the macroblock

DWord	Bit	Description
	15:12	**Reserved**
	12:0	**MbHdrBits** : Header Bit count (bit count due to Pre-coefficient data) for the macroblock
2	31:27	**Reserved**
	26:0	**Cbp:** Coded Block Pattern of sub-blocks
3	31:30	**Reserved**
	29	**IntraMBFlag**
	28:24	**MBType5Bits**
	23:17	**Reserved**
	16	**ClampFlag**: Coefficient clamping flag for RC (Status) 1 - Indicates if clamping of any coefficient is done on the macroblock for Rate Control
	15:0	**Reserved (future QRC stat output)**

6.1 PAK Multi-Pass

Multi-Pass PAK Usages:

- Intra MB 3200-bit conformance
- Inter MB Re-quantization
- Frame level Re-quantization

How to Enable Multi-Pass PAK?

- Using the existing conditional batch buffer execution capability to skip/execute the second pass
    - How to dynamically change the condition?
        - Defined one error condition register with a mask. Do HW status page update at the end of the first pass. 0 means all OK, non-zero means there is an error condition, requiring second pass. Mask is used by the host to control what kind of multi-pass is intended.
        - For example, one error bit is 3200-bit conformance violation. Another error bit is the total bit count exceeds (too much or too little) the target range (need to define the target range in the state).
        - **The logic pefectly fits in the conditional batch buffer control logic that VCS has today in GT. There is no additional logic need to be added in VCS to support media functionality.** (**Batch Buffer Skip:** This field only takes effect if Compare Semaphore is set and the value at **Semaphore Address is NOT greater than the Semaphore Data Dword)**.
- Adding a picture level state command to enable and control the behavior of the second pass PAK

- How to control the re-PAK? Added 3 conformance flags (error registers) in the per-MB streamout. Then the error control is based on the error register and the mask defined in picture level states. There are 8 register flags defined out of which only the 3200-bit case has usage model defined for today. The rest are left for future usage.

Issues and Limitations:

- There is no programmable engine in MFX for flexible control: Therefore, whatever we have defined must consider flexibility

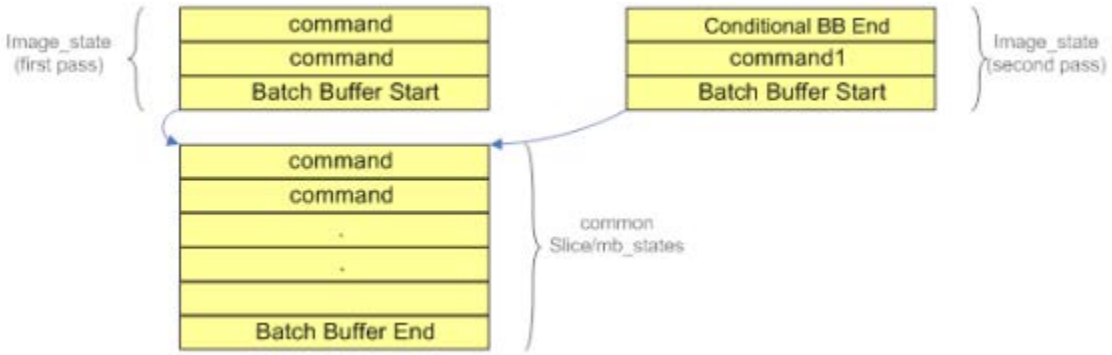
Following 2 MI packets are used inside VCS without any change to support Multipass-PAK behaviour.

- MI_Conditional_Batch_Buffer_End
- Memory Interface Registers

6.2 Driver Usage

Driver places Image states in one batch buffer and all slice level and macroblock level states into another batch buffer and link 2 batch buffers. Also replicate Image states with multipass changes in another batch buffer link them to slice/macroblock batch buffer. In this way, only Image states are replicated but not the slice/macroblock states. The image states includes all buffers defined at image(indirectMV, original pixel buffer, etc). Following changes are needed in the Multipass Image State,

- **Reset- Stream-Out Enable(disable stream out in the second pass)**
- **Set- MacroblockStatEnable (enable reading of macroblock status buffer)**
- **Reset- 3200-bit conformance (do not report 3200-bit conformance)**



- **Define Conditional Batch Buffer End for CS/VCSVINunit**

7. Programming Reference

7.1 Monochrome Picture Processing

Monochrome picture is specified using the Surface State with Surface Format of 12. Therefore, MFX hardware, in either decode or encode mode, doesn't generation any read or write traffic for U/V components. *Motivation for this bandwidth optimization is that monochrome video coding might be used for wireless display.*

For Encoder :

1. No read in UV original components.
2. processing UV component - no
3. reconstructed UV component reference picture - no
4. filter UV component - no

For Decoder :

1. VLD mode : no color component coming out in Monochrome mode and so no processing and not writing output
2. IT mode : there is no color component in the coefficient buffer, and so no processing and not writing output

7.2 Context Switch

There is no pre-emption for the BCS pipeline; hence every command buffer is required to contain all the states setup (preamble). Specifically, CPU can not interrupt the BCS-BSD pipe, to stop the operation in the middle of decoding a bitstream data.

Switch of contexts can only be performed at picture boundary.

No state needs to be saved.

7.3 Pipeline Flush

Implicit flush for AVC and VC1 is performed at the end of Slice : for MPEG2 is done when a new image/picture command is issued. Because MPEG2 a slice can be one MB, no point to flush. MPEG2 will snoop the next command if it is an img_state command.

Explicit flush MI (1 bit to do media pipeline vs Gx pipeline) flush and cache flush (switch reference frame) – MI flush has bit to do cache flush. MI flush is for driver synchronization.

7.4 MMIO Interface

A set of registers are defined and accessible through MMIO interface to serve multiple purposes:

- Use for system configuration
- For accessing Performance counters

Register Name	Description	Register Type	Address Offset	Dec/Enc
MFD ERROR Status	MFD ERROR STATUS_VLD ERROR flags and counter	RO	12400	Dec
Reserved	MBZ		12404~1241C	
MFD picture-level parameter	VC1 picture level parameters	R/W	12420	Dec
Reserved	MBZ		12434	
MFX PIPELINE_STATUS_FLAGS	MFX PIPELINE STATUS Flags_MFX pipeline mode flags	RO	12438	Dec
MFX_Error_Injection_Parameter	Control HW error injector	WO	12454	Dec
Reserved			12458~1245C	
MFX Frame Performance count	Number of clocks spent decoding/encoding a frame	RO	12460	Dec/Enc
MFX Slice Performance count	Number of clocks spent decoding/encoding a slice	RO	12464	Dec/Enc
MFX Frame Macroblock count	Number of MBs decoded/encoded per frame	RO	12468	Dec/Enc
MFD Frame BITSTREAM SE/BIN count	Number of bin/SE decoded per frame	RO	1246C	Dec
MFX Memory Latency count1	Reference picture read latency -min and max	RO	12470	Dec/Enc
MFX Memory Latency count2	Reference picture read latency - Accumulative (used for compute AVE latency)	RO	12474	Dec/ENc
MFX Memory Latency count3	row-store/bit-stream memory read latency -min and max	RO	12478	Dec/Enc
MFX Memory Latency count4	row-store/bit-stream memory read latency - accumulative (used to compute AVE latency)	RO	1247C	Dec/End
MFX Frame row-stored/bit-stream read Count	# of row-store memory requests sent	RO	12480	Dec/End
MFX Motion Comp read Count	total number of CL memory accesses per frame	RO	12484	Dec/ENd
MFX Motion Comp MISS Count	total number of CL HITs per frame	RO	12488	Dec/ENd
Reserved			1248C~1249C	
MFC_BITSTREAM_BYTECOUNT_FRAME	Total Bitstream Output Byte Count register per Frame	RO	124A0	Enc
MFC_BITSTREAM_SE_BITCOUNT_FRAME	Bitstream Output total Byte Count for syntax eements (total byes of MB data from SEC per frame)	RO	124A4	Enc
MFC_AVC_CABAC_BIN_COUNT_FRAME	Bitstream Output total bin count per frame	RO	124A8	Enc
MFC_AVC_CABAC_INSERTION_COUNT	Bitstream Output CABAC Insertion Count Register	RO	124AC	Enc
MFC_AVC_MINSIZE_PADDING_COUNT	Bitstream Output Minimal Size Padding Count Register	RO	124B0	Enc
MFC_IMAGE_STATUS_MASK	image status(flags).	R/W	124B4	Enc
MFC_IMAGE_STATUS_CONTROL	suggested data for next frame in multi-pass.	RO	124B8	Enc
MFC_QP_STATUS_COUNT	Overall adjusted delta QP via multi-pass, Sum of QPY for all macroblocks of the frame	RO	124BC	Enc
			124C0~124CC	Enc
MFC_BITSTREAM_BYTECOUNT_SLICE	Bitstream Output Byte Count	RO	124D0	Enc

Register Name	Description	Register Type	Address Offset	Dec/Enc
	Register per Slice			
MFC_BITSTREAM_SE_BITCOUNT_SLICE	Bitstream Output Bit Count for the last Syntax Element Register	RO	124D4	Enc
PAK_ REPORT_WARNING	MPC Warning Register	RO	124E4	Enc
PAK_REPORT_ERROR	MPC Error Register	RO	124E8	Enc
PAK_REPORT_RUNNING	PAK_REPORT_RUNNING status register	RO	124EC	Enc
Reserved			124F0~124FC	Enc

7.4.1 Decoder Registers

7.4.1.1 MFD ERROR STATUS_VLD ERROR flags and counter

MFD_ERROR_STATUS - MFD Error Status

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	12400h

This register stores the error status flags and count reports by the bit-stream decoder.
This register is not part of hardware context save and restore. Driver should read the content prior to starting a new batch/frame.

DWord	Bit	Description
0	31:16	**Number of Error Events** Exists If: JPEG == True Format: U16 This 16-bit field indicates the number of error events detected during decoding the current frame. This field is clear at the start of decoding a new frame.
	31:16	**Number of MB Concealment** Exists If: AVC CAVLC, AVC CABAC, VC1 and MPEG2 == True Format: U16 This 16-bit field indicates the number of MB is concealmed by hardware. This field is clear at the start of decoding a new frame.
	15:0	**Bit-stream Error flags** Bit-stream error detected by the VLD bit-steram decoder. These flags are reset at the beginning of a frame and updated until starting of another frame. AVC CAVLC: Please refer to AVC CAVLC table for each bit field AVC CABAC: Please refer to AVC CABAC table for each bit field VC1: Please refer to VC1 table for each bit field MPEG2: Please refer to MPEG2 table for each bit field JPEG: Please refer to JPEG table for each bit field

7.4.1.2 AVC CAVLC

AVC CAVLC		
Source:		VideoCS
Default Value:		0x00000000
DWord	**Bit**	**Description**
0	15	**Total Zero out-of-bound Error** This flag indicates the Total zero SE count exceed the max number of coeffs allowed in an intra16x16 AC block.
	14	**Coefficient level out-of-bound Error** This flag indicates the coded coefficient level SEs in the bit-stream is out-of-bound.
	13	**RunBefore out-of-bound Error** This flag indicates the coded RunBefore SE value is larger than the remaining zero block count.
	12	**Total coefficient Out-of-bound Error** This flag indicates the coded total coeff SE count exceed the max number of coeffs allowed in an intra16x16 AC block.
	11	**Temporal Direction Motion Vector Out-of-Bound Error** This flag indicates motion vectors calculated from Temporal Direct Motion Vector is larger than the allowed range specified by the AVC spec.
	10	**Final Motion Vector Out-of-Bound Error** This flag indicates final reconstructed Motion Vector value is larger than the allowed range specified by the AVC spec.
	9	**Motion Vector Delta SE Out-of-Bound Error** This flag indicates inconsistent Motion Vector Delta SEs coded in the bit-stream.
	8	**Reference Index SE Out-of-Bound Error** This flag indicates inconsistent Reference Index SEs coded in the bit-stream.
	7	**RunBefore/TotalZero Error** This flag indicates one or more inconsistent RunBefore or TotalZero SEs coded in the bit-stream.
	6	**Exponential Golomb Error** This flag indicates hardware detects more than 18 leadzero for skip and more than 19 for other SEs from the Exponential Golomb Logic
	5	**Total Coeff SE Error** This flag indicates one or more inconsistent total coeff SEs coded in the bit-stream.
	4	**Macroblock Coded Block Pattern Error** This flag indicates inconsistent CBP SEs coded in the bit-stream.
	3	**Mbytpe/submbtype Error** This flag indicates inconsistent MBtype/SubMBtype SEs coded in the bit-stream.
	2	**Chroma Intra prediction Mode Error** This flag indicates inconsistent Chroma Intra prediction mode SEs coded in the bit-stream.
	1	**Luma Intra prediction Mode Error** This flag indicates inconsistent luma Intra prediction mode SE coded in the bit-stream.
	0	**MB Concealment Flag** Each pulse from this flag indicates one MB is concealed by hardware.

7.4.1.3 AVC CABAC

AVC CABAC

Source: VideoCS

Default Value: 0x00000000

DWord	Bit	Description
0	15	**Reserved** Format: MBZ
	14	**Coefficient level out-of-bound Error** This flag indicates the coded coefficient level SEs in the bit-stream is out-of-bound.
	13	**Reserved** Format: MBZ
	12	**Reserved** Format: MBZ
	11	**Temporal Direction Motion Vector Out-of-Bound Error** This flag indicates motion vectors calculated from Temporal Direct Motion Vector is larger than the allowed range specified by the AVC spec.
	10	**Final Motion Vector Out-of-Bound Error** This flag indicates final reconstructed Motion Vector value is larger than the allowed range specified by the AVC spec.
	9	**Motion Vector Delta SE Out-of-Bound Error** This flag indicates inconsistent Motion Vector Delta SEs coded in the bit-stream.
	8	**Reference Index SE Out-of-Bound Error** This flag indicates inconsistent Reference Index SEs coded in the bit-stream.
	7	**MacroBlock QpDelta Error** This flag indicates out-of-bound MB QP delta SEs coded in the bit-stream.
	6	**Motion Vector Delta SE Error** This flag indicates out-of-bound motion vector delta SEs coded in the bit-stream.
	5	**Reference Index SE Error** This flag indicates out-of-bound Refidx SEs coded in the bit-stream.
	4	**Residual Error** This flag indicates out-of-bound absolute coefficient level SEs coded in the bit-stream.
	3	**Slice end Error** This flag indicates a pre-matured slice_end SE or inconsistent slice end on the last MB of a slice.
	2	**Chroma Intra prediction Mode Error** This flag indicates inconsistent Chroma Intra prediction mode SEs coded in the bit-stream.
	1	**Luma Intra prediction Mode Error** This flag indicates inconsistent luma Intra prediction mode SE coded in the bit-stream.
	0	**MB Concealment Flag** Each pulse from this flag indicates one MB is concealed by hardware.

7.4.1.4 VC1

VC1		
Source:	VideoCS	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	15:8	**Reserved** Format: MBZ
	7	**Syncmarker Error** This flag indicates missing sync marker SEs coded in the bit-stream.
	6	**Mbmode SE Error** This flag indicates inconsistent Macroblock SEs coded in the bit-stream.
	5	**Transformtype SE Error** This flag indicates inconsistent transform type SEs coded in the bit-stream.
	4	**Coefficient Error** This flag indicates inconsistent Coefficient SEs coded in the bit-stream.
	3	**Motion Vector SE Error** This flag indicates inconsistent Motion Vector SEs coded in the bit-stream.
	2	**Coded Block Pattern CY SE Error** This flag indicates inconsistent CBPCY SEs coded in the bit-stream.
	1	**Mquant Error** This flag indicates inconsistent MQUANT SEs coded in the bit-stream.
	0	**MB Concealment Flag** . Each pulse from this flag indicates one MB is concealed by hardware.

7.4.1.5 MPEG2

MPEG2		
Source:	VideoCS	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	15:6	**Reserved** Format: MBZ
	5	**Missing EOB Error** This flag indicates missing EOB SEs coded in the bit-stream. Missing EOBs are concealed to match CBP of the error MB.
	4	**Inconsistent starting position Error – overlapping MBs** This flag indicates two slices overlapping one another by one or more MBs. Duplicate MBs decoded off the second slice shall be discarded.
	3	**Slice out-of-bound Error** This flag indicates a slice is running beyond the width of the picture. Out-of-bound MBs shall be discarded.
	2	**Premature frame end Error** This flag indicates missing slices/MBs coded in the bit-stream of a frame. One or more MBs are

MPEG2		
		concealed to reach end of picture.
	1	**Inconsistent starting position Error – Missing MBs** This flag indicates one or more MBs are being concealed due to inconsistent MB starting and ending positions between slices.
	0	**MB Concealment Flag** . Each pulse from this flag indicates one MB is concealed by hardware.

7.4.1.6 JPEG

JPEG		
Source:		VideoCS
Default Value:		0x00000000
DWord	**Bit**	**Description**
0	15:5	**Reserved** Format: MBZ
	4	**Inconsistent VLD SE Error** This flag indicates an inconsistent SE coded in the bit-stream. Bit-stream does not match any entries in the hauffman table.
	3	**Extra Block Error** This flag indicates extra block coded within an ECS data boundary.
	2	**Missing block Error** This flag indicates one or more blocks are missing within an ECS data boundary.
	1	**Extra ECS Error** This flag indicates extra ECS' coded in the bit-stream SCAN payload data.
	0	**Missing ECS Error** This flag indicates one or more ECS' are missing from the bit-stream SCAN payload data.

7.4.1.7 MFD PICTURE PARAMETER - MFD Picture Parameter

MFD_PICTURE_PARAM - MFD Picture Parameter		
Register Space:	MMIO: 0/2/0	
Source:	VideoCS	
Default Value:	0x00000000	
Access:	R/W	
Size (in bits):	32	
Trusted Type:	1	
Address:	12420h	

DWord	Bit	Description
0	31:0	**Reserved** Format: MBZ

7.4.1.8 MFX PIPELINE STATUS Flags_MFX pipeline mode flags

MFX_STATUS_FLAGS - MFX Pipeline Status Flags		
Register Space:	MMIO: 0/2/0	
Source:	VideoCS	
Default Value:	0x00000000	
Access:	RO	
Size (in bits):	32	
Trusted Type:	1	
Address:	12438h	

This register stores the various pipeline status flags.
This register is not part of hardware context save and restore.

DWord	Bit	Description
0	31:17	**Reserved** Format: MBZ
	16	**MFX Active** Frame decoding/encoding is in progress. Set on frame_start; clear on frame_end.
	15:10	**Reserved** Format: MBZ
	9	**Streamout Enable**
	7	**Post Deblocking Mode Enable**
	6	**Pre Deblocking Mode Enable**

MFX_STATUS_FLAGS - MFX Pipeline Status Flags

	5	**Decoder Mode Select** Value: 0 — Configure the MFD Engine for VLD Mode Value: 1 — Configure the MFD Engine for IT Mode
	4	**Codec Select** Value: 0 — Decode Value: 1 — Encode
	3:2	**Video Mode** Value: 00b — MPEG2 Value: 01b — VC1 Value: 10b — AVC Value: 11b — JPEG
	1	**Decoder Short Format Mode** Value: 0 — AVC/VC1 Short Format Mode is in use Value: 1 — AVC/VC1 Long Format Mode is in use
	0	**Stitch Mode** Value: 0b — Not in Stitch Mode Value: 1b — In the Special Stitch Mode

7.4.1.9 MFX_FRAME_PERFORMANCE_CT - MFX Frame Performance Count

MFX_FRAME_PERFORMANCE_CT - MFX Frame Performance Count

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	12460h

This register stores the number of clock cycles spent decoding/encoding the current frame.
This register is not part of hardware context save and restore.

DWord	Bit	Description
0	31:0	**MFX Frame Performance Count** Total number of clocks between frame start and frame end. This count is incremented on crm_clk

7.4.1.10 MFX Slice Performance Count – Reported clock count per slice

MFX_SLICE_PERFORM_CT - MFX Slice Performance Count	
Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	12464h

This register stores the number of clock cycles spent decoding/encoding the current slice.
This register is not part of hardware context save and restore.

DWord	Bit	Description
0	31:0	**MFX Frame Performance Count** Total number of clocks between slice start and slice end. This count is incremented on crm_clk

7.4.1.11 MFX_MB_COUNT - MFX Frame Macroblock Count

MFX_MB_COUNT - MFX Frame Macroblock Count	
Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	12468h

This register stores the number of Macro-blocks decoded/encoded in current frame.
This register is not part of hardware context save and restore.

DWord	Bit	Description
0	31:0	**MFX Frame Macro-block Count** Total number of Macro-block decoded/encoded in current frame. This number is used with frame performance count to derive clk/mb.

7.4.1.12 MFX_SE-BIN_CT - MFX Frame BitSteam SE/BIN Count

MFX_SE-BIN_CT - MFX Frame BitStream SE/BIN Count	
Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	1246Ch

This register stores the number of BINs (AVC CABAC) and SEs (CAVLD, VLD) decoded in a frame.
This register is not part of hardware context save and restore.

DWord	Bit	Description
0	31:0	**MFX Frame Bit-stream SE/BIN Count** Total number of BINs/SEs decoded in current frame. This number is used with frame performance count to derive Bin/clk or SE/clk.

7.4.1.13 MFX Memory Latency Count1 – Reported Reference read latency Counts

MFX_LAT_CT1 - MFX_Memory_Latency_Count1	
Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	12470h

This register stores the max and min memory latency counts reported on reference read requests.
This register is not part of hardware context save and restore.

DWord	Bit	Description
0	31:24	**Max Request Count** This field indicates the maximum number of requests allowed by the memory sub-system channel.
	23:16	**Current Request Count** This field indicates the number of requests currently outstanding in the memory sub-system. This field should report with a value of zero at the end of frame; otherwise the motion compensation engine is most likely hung waiting for read data to be returned from sub-system.
	15:8	**MFX Reference picture read request - Max Latency Count in 8xMedia clock cycles** This field reports the maximum memory latency count on all reference reads requested by the motion compensation engine.

MFX_LAT_CT1 - MFX_Memory_Latency_Count1		
	7:0	**MFX Reference picture read request - Min Latency Count in 8xMedia clock cycles** This field reports the minimum memory latency count on all reference reads requested by the motion compensation engine.

7.4.1.14 MFX_LAT_CT2 - MFX Memory Latency Count2

MFX_LAT_CT2 - MFX Memory Latency Count2	
Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	12474h

This register stores the accumulative memory latency count on reference picture read requests.
This register is not part of hardware context save and restore.

DWord	Bit	Description
0	31:26	**Reserved** Format: MBZ
	25:0	**MFX Reference picture read request - Accumulative Memory Latency Count for the entire frame in 8xMedia clock cycles** The accumulative memory latency count of all reference reads requested by motion compensative engine per frame. This number is used with **MFX Frame Motion Comp Read Count** to derive average memory latency.

7.4.1.15 MFX_LAT_CT3 - MFX Memory Latency Count3

MFX_LAT_CT3 - MFX Memory Latency Count3	
Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	12478h

This register stores the max and min memory latency counts reported on row-stored/bit-stream read requests. Max

MFX_LAT_CT3 - MFX Memory Latency Count3

and current requests into memory sub-system engine.
This register is not part of hardware context save and restore.

DWord	Bit	Description
0	31:24	**Max Request Count** This field indicates the maximum number of requests allowed by the memory sub-system channel.
	23:16	**Current Request Count** This field indicates the number of requests currently outstanding in the memory sub-system. This field should report with a value of zero at the end of frame; otherwise the pre-fetch engine most likely hung waiting for read data to be returned from sub-system.
	15:8	**MFX row-stored/bit-stream read request - Max Latency Count in 8xMedia clock cycles** This field reports the maximum memory latency count on all row-stored/bit-stream reads requested by the memory pre-fetch engine.
	7:0	**MFX row-stored/bit-stream read request - Min Latency Count in 8xMedia clock cycles** This field reports the minimum memory latency count on all row-stored/bit-stream reads requested by the memory pre-fetch engine.

7.4.1.16 MFX_LAT_CT4 - MFX Memory Latency Count4

MFX_LAT_CT4 - MFX Memory Latency Count4

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	1247Ch

This register stores the accumulative memory latency count on row-stored/bit-stream read requests.
This register is not part of hardware context save and restore.

DWord	Bit	Description
0	31:26	**Reserved** Format: MBZ
	25:0	**MFX row-stored/bit-stream read request - Accumulative Memory Latency Count for the entire frame in 8xMedia clock cycles** The accumulative memory latency count of all row-stored/bit-stream reads requested by pre-fetch engine per frame. This number is used with **Frame row-stored/bit-stream memory read count** to derive average memory latency.

7.4.1.17 MFX_SE-BIN_CT - MFX Frame BitStream SE/BIN Count

MFX_SE-BIN_CT - MFX Frame BitStream SE/BIN Count

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	1246Ch

This register stores the number of BINs (AVC CABAC) and SEs (CAVLD, VLD) decoded in a frame.
This register is not part of hardware context save and restore.

DWord	Bit	Description
0	31:0	**MFX Frame Bit-stream SE/BIN Count** Total number of BINs/SEs decoded in current frame. This number is used with frame performance count to derive Bin/clk or SE/clk.

7.4.1.18 MFX_READ_CT - MFX Frame Motion Comp Read Count

MFX_READ_CT - MFX Frame Motion Comp Read Count

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	12484h

This register stores the total number of reference picture read requests made by the Motion Compensation engine per frame.
This register is not part of hardware context save and restore.

DWord	Bit	Description
0	31:20	**Reserved** Format: MBZ
	19:0	**MFX Frame Motion Comp CL read request Count** Total number of reference picture read requests by the motion compensation engine per frame.

7.4.1.19 MFX_READ_CT - MFX Frame Motion Comp MISS Count

MFX_MISS_CT - MFX Frame Motion Comp Miss Count	
Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	12488h

This register stores the total number of cacheline hits occurred in the motion compensation cache per frame. This register is not part of hardware context save and restore.

DWord	Bit	Description
0	31:16	**Reserved** Format: MBZ
	15:0	**MFX Frame Motion Comp cache miss Count** Total number of CL misses occurred in the 12KB cache of the motion compensation engine per frame. This number is used along with **MFX Frame Motion Comp Read Count** to derive motion comp cache miss/hit ratio.

7.4.2 Encoder Registers

7.4.2.1 MFC_VIN_AVD_ERROR_CNTR — AVC Bitstream Decoding Front-End Parsing Logic Error Counter Report Register

MFC_VIN_AVD_ERROR_CNTR - MFC_AVC Bitstream Decoding Front-End Parsing Logic Error Counter	
Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	12804h

DWord	Bit	Description
0 avd_error_flagsR[31:0]	31:0	**Reserved** Format: MBZ

7.4.3 MFC_BITSTREAM_BYTECOUNT_FRAME — Reported Bitstream Output Byte Count per Frame

MFC_BITSTREAM_BYTECOUNT_FRAME - Reported Bitstream Output Byte Count per Frame Register	
Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	124A0h

This register stores the count of bytes of the bitstream output per frame

DWord	Bit	Description
0	31:0	**MFC Bitstream Byte Count per Frame** Total number of bytes in the bitstream output per frame from the encoder. This includes header/tail/byte alignment/data bytes/EMU (emulation) bytes/cabac-zero word insertion/padding insertion. This count is updated for every time the internal bitstream counter is incremented and its reset at image start.

7.4.4 MFC_BITSTREAM_SE_BITCOUNT_FRAME (Reported Bitstream Output Bit Count for Syntax Elements Only)

MFC_BITSTREAM_SE_BITCOUNT_FRAME - Reported Bitstream Output Bit Count for Syntax Elements Only Register	
Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	124A4h

This register stores the count of number of bits in the bitstream due to syntax elements only. This excludes header/ byte alignment /tail/EMU/CABAC-0word/padding bits but includes the stop-one-bit. This register is part of the context save and restore.

DWord	Bit	Description
0	31:0	**MFC Bitstream Syntax Element Only Bit Count** Total number of bits in the bitstream output due to syntax elements only. It includes the data bytes only. This count is updated for every time the internal bitstream counter is incremented and its reset at image start.

7.4.5 MFC_AVC_CABAC_BIN_COUNT_FRAME (Reported Bitstream Output CABAC Bin Count)

MFC_AVC_CABAC_BIN_COUNT_FRAME - Reported Bitstream Output CABAC Bin Count Register	
Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	124A8h

This register stores the count of number of bins per frame.

DWord	Bit	Description
0	31:0	**MFC AVC Cabac Bin Count** Total number of BINs in the bitstream output per frame from the encoder. This count is updated for every time the bin counter is incremented and its reset at image start.

7.4.6 MFC_AVC_CABAC_INSERTION_COUNT — Reported Bitstream Output CABAC Insertion Count

AVC_CABAC_INSERTION_COUNT - MFC_AVC_CABAC_INSERTION_COUNT	
Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	124ACh

This register stores the count in bytes of **CABAC ZERO_WORD insertion. It is primarily provided for statistical data gathering.**

DWord	Bit	Description
0	31:0	**MFC AVC Cabac Insertion Count** Total number of bytes in the bitstream output before for the CABAC zero word insertion. This count is updated each time when the insertion count is incremented.

7.4.7 MFC_AVC_MINSIZE_PADDING_COUNT — Reported Bitstream Output Minimal Size Padding Count

MFC_AVC_MINSIZE_PADDING_COUNT - Bitstream Output Minimal Size Padding Count Report Register	
Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	12414h
Name:	VDBOX1

This register stores the count in bytes of **minimal size padding insertion. It is primarily provided for statistical data gathering.** This register is part of the context save and restore.

DWord	Bit	Description
0	31:0	**MFC AVC MinSize Padding Count** Total number of bytes in the bitstream output contributing to minimal size padding operation. This count is updated each time when the padding count is incremented.

7.5 MFC_IMAGE_STATUS_MASK

MFC_IMAGE_STATUS_MASK - MFC Image Status Mask	
Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	124B4h

This register stores the image status(flags).

DWord	Bit	Description
0	31:0	**Control Mask** Control Mask for dynamic frame repeat.

7.5.1 MFC_IMAGE_STATUS_CONTROL

MFC_IMAGE_STATUS_CONTROL - MFC Image Status Control	
Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Trusted Type:	1
Address:	124B8h

This register stores the suggested data for next frame in multi-pass.

DWord	Bit	Description
0	31:24	**Cumulative slice delta QP**
	23:16	**QP Value** suggested slice QP delta value for frame level Rate control. This value can be +ve or -ve
	15	**QP-Polarity Change** Cumulative slice delta QP polarity change.
	14:13	**Num-Pass Polarity Change** Number of passes after cumulative slice delta QP polarity changes.
	12	**Reserved** Format: MBZ
	11:8	**Total Num-Pass**
	7:3	**Reserved** Format: MBZ

MFC_IMAGE_STATUS_CONTROL - MFC Image Status Control		
	2	**Panic** Panic triggered to avoid too big packed file.
	1	**Frame Bit Count** Frame Bit count over-run/under-run flag
	0	**Max Conformance Flag** Max Macroblock conformance flag or Frame Bit count over-run/under-run

7.5.2 MFC_QUP_CT - MFC QP Status Count

MFC_QUP_CT - MFC QP Status Count			
Register Space:		MMIO: 0/2/0	
Source:		VideoCS	
Default Value:		0x00000000	
Access:		RO	
Size (in bits):		32	
Trusted Type:		1	
Address:		124BCh	
This register stores the suggested QP COUNTS in multi-pass.			
DWord	**Bit**	**Description**	
0	31:24	**Cumulative QP Adjust**	
		Format:	U8
		Cumulative QP adjustment after multiple passes. If there is no need to multi-pass, this value would be zero. (This is in sign magnitude form).	
	23:0	**Cumulative QP**	
		Format:	U24
		Cumulative QP for all MB of a Frame (Can be used for computing average QP).	

7.5.3 MFC_BITSTREAM_BYTECOUNT_SLICE — Bitstream Output Byte Count per Slice

MFC_BITSTREAM_BYTECOUNT_SLICE - Bitstream Output Byte Count Per Slice Report Register	
Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	124D0h

This register stores the count of bytes of the bitstream output. This register is part of the context save and restore.

DWord	Bit	Description
0	31:0	**MFC Bitstream Byte Count** Total number of bytes in the bitstream output from the encoder. This count is updated for every time the internal bitstream counter is incremented.

7.5.4 MFC_BITSTREAM_SE_BITCOUNT_SLICE — Bitstream Output Bit Count for the last Syntax Element

MFC_BITSTREAM_SE_BITCOUNT_SLICE - Bitstream Output Bit Count for the last Syntax Element Report Register	
Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	124D4h
Name:	VDBOX1

This register stores the count of number of bits in the bitstream for the last syntax element before padding. The bit count is before the byte-aligned alignment padding insertion, but includes the stop-one-bit. This register is part of the context save and restore.

DWord	Bit	Description
0	31:0	**MFC Bitstream Syntax Element Bit Count** Total number of bits in the bitstream output before padding. This count is updated each time the internal counter is incremented.

7.6 Row Store Sizes and Allocations

	AVC	VC1	MPEG2	JPEG	IT	ENC	SEC ENC
vin_vmx_pixcoefind_ addr[31:6]	Bitstream	Bitstream	Bitstream	Bitstream	VDS COEF	Orig Pix	BSP data
vin_vmx_mvbsdrs_ addr[31:6]	VAD BSD		VMD RS		VDS MV	MPC MV	
vin_vmx_mpcildbmpr_ addr[31:6]	VAM MPR				VDS ILDB	MPC RS	
vin_vmx_dmv*_ addr[31:6]	VAM DMV	VCP DMV					
vin_vmx_bp_addr [31:0]		VCP BP					

	Write		Surf size
	VBP BP	vin_bp_addr	Frame width/pitch * Height
	VMD RS	vin_vmx_mvbsdrs_addr	Frame width
	VCP RS	vin_vmx_mvbsdrs_addr	Frame width
	VCP DMV	vin_vmx_dmv1_addr	Frame size
	VAD BSD	vin_vmx_mvbsdrs_addr	Frame width * (1+mbaff)
	VAM MPR	vin_vmx_mpcildbmpr_addr	Frame width * (1+mbaff)
	VAM DMV	34x1 mux, from IDC	Frame size
	Streamout	vin_streamout_addr	Frame size
	VOP RS	vin_ipred_os_addr	Frame width
	MPC RS	vin_vmx_mpcildbmpr_addr	Frame width * (1+mbaff)
	BSP BS	Direct from BSP	
	BSP MB	Direct from BSP	
	Read		
row store	VMD	**vin_vmx_mvbsdrs_addr**	Frame width
row store	VCP	**vin_vmx_mvbsdrs_addr**	Frame width
DMV	VCP	**vin_vmx_dmv*_addr**	Frame size
Bitplane	VCD	**vin_vmx_bp_addr**	Frame width/pitch * Height
Bsd	VAD	**vin_vmx_mvbsdrs_addr**	Frame width * (1+mbaff)
Mpr	VAM	**vin_vmx_mpcildbmpr_addr**	Frame width * (1+mbaff)
Dmv	VAM	**vin_vmx_dmv*_addr**	Frame size
Coef	VDS	**vin_vmx_pixcoefind_addr**	Obj

	Write		**Surf size**
Mv	VDS	**vin_vmx_mvbsdrs_addr**	Obj
Ildb	VDS	**vin_vmx_mpcildbmpr_addr**	Obj
Rs	VIP	**vin_ipred_os_addr**	Frame width
RS	MPC	**vin_vmx_mpcildbmpr_addr**	Frame width * (1+mbaff)
MV	MPC	**vin_vmx_mvbsdrs_addr**	Obj
sec enc	BSP	**vin_vmx_mvbsdrs_addr**	Obj
multipass	VIN	**vin_vmx_bp_addr**	Frame size
orig pix	USB	**vin_vmx_pixcoefind_addr**	Frame size

MPEG2 VLD Decoding Mode :

use BSD Row Store only, and

MPEG2 IT Decoding Mode :

MPEG2 IT mode does not need row-store

JPEG VLD Decoding Mode : no row store is needed

Revision History

Revision Number	Description	Revision Date
1.0	First 2012 OpenSource edition	2012
1.1	Revisions based on user feedback	2012

§§