Intel® Open Source HD Graphics

Programmer's Reference Manual

For the 2016 Intel Atom™ Processors, Celeron™ Processors, and Pentium™ Processors based on the "Apollo Lake" Platform (Broxton Graphics)

Volume 2a: Command Reference: Instructions (Command Opcodes)

May 2017, Revision 1.0

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.

Notices and Disclaimers

Table of Contents

3DSTATE_DS

3DSTATE_DS

Source: RenderCS

Length Bias: 2

The state used by DS is defined with this inline state packet

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 1Dh 3DSTATE_DS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 9h Excludes DWord (0,1) Format: =n Total Length - 2
1..2	63:6	**Kernel Start Pointer** Format: InstructionBaseOffset[63:6]Kernel This field specifies the starting location of the kernel program run by threads spawned by this FF unit. It is specified as a 64-byte-granular offset from the Instruction Base Address. This field is ignored if DS Function Enable is DISABLED.
	5:0	**Reserved** Format: MBZ
3	31	**Reserved** Format: MBZ

3DSTATE_DS

30 Vector Mask Enable

Format:	U1 Enumerated Type

Upon subsequent DS thread dispatches, this bit is loaded into the EU's Vector Mask Enable (VME, cr0.0[3]) thread state. Refer to EU documentation for the definition and use of VME state.

Value	Name	Description
0h	Dmask	The EU will use the Dispatch Mask (supplied by the DS stage) for instruction execution.
1h	Vmask	The EU will use the Vector Mask (derived from the Dispatch Mask) for instruction execution.

Programming Notes

Under normal conditions SW shall specify DMask, as the DS stage will provide a Dispatch Mask appropriate to SIMD4x2 or SIMD8 thread execution (as a function of dispatch mode). E.g., for SIMD4x2 thread execution, the DS stage will generate a Dispatch Mask that is equal to what the EU would use as the Vector Mask. For SIMD8 execution there is no known usage model for use of Vector Mask (as there is for PS shaders).

29:27 Sampler Count

Format:	U3

Specifies how many samplers (in multiples of 4) the kernel uses. Used only for prefetching the associated sampler state entries.
This field is ignored if DS Function Enable is DISABLED.

Value	Name	Description
0h	No Samplers	No samplers used
1h	1-4 Samplers	between 1 and 4 samplers used
2h	5-8 Samplers	between 5 and 8 samplers used
3h	9-12 Samplers	between 9 and 12 samplers used
4h	13-16 Samplers	between 13 and 16 samplers used

26 Reserved

Format:	MBZ

25:18 Binding Table Entry Count

Format:	U8

When HW Generated Binding Table is disabled:
Specifies how many binding table entries the kernel uses. Used only for prefetching of the binding table entries and associated surface state.
Note: For kernels using a large number of binding table entries, it may be wise to set this field to zero to avoid prefetching too many entries and thrashing the state cache.
This field is ignored if DS Function Enable is DISABLED.

3DSTATE_DS

When HW Generated Binding Table bit is enabled:
This field indicates which cache lines (512bit units - 32 Binding Table Entry section) should be fetched. Each bit in this field corresponds to a cache line. Only the 1st 4 non-zero Binding Table entries of each 32 Binding Table entry section prefetched will have its surface state prefetched.

Value	Name
[0,255]	

Programming Notes
When HW binding table bit is set, it is assumed that the Binding Table Entry Count field will be generated at JIT time.

17 **Thread Dispatch Priority**

Format:	U1 Enumerated Type

Specifies the priority of the thread for dispatch:
This field is ignored if DS Function Enable is DISABLED.

Value	Name	Description
0h	Normal	Normal Priority
1h	High	High Priority

16 **Floating Point Mode**

Format:	U1 Enumerated Type

Specifies the initial floating point mode used by the dispatched thread.
This field is ignored if DS Function Enable is DISABLED.

Value	Name	Description
0h	IEEE-754	Use IEEE-754 Rules
1h	Alternate	Use alternate rules

15 **Reserved**

Format:	MBZ

14 **Accesses UAV**

Format:	Enable

This bit gets loaded into EU CR0.1[12] (note the bit # difference). See Exceptions and ISA Execution Environment.

Programming Notes
This field must not be set when DS Function Enable is disabled.

13 **Illegal Opcode Exception Enable**

Format:	Enable

This bit gets loaded into EU CR0.1[12] (note the bit # difference). See Exceptions and ISA Execution Environment.
This field is ignored if DS Function Enable is DISABLED.

3DSTATE_DS

	12:8	**Reserved** Format: MBZ
	7	**Software Exception Enable** Format: Enable This bit gets loaded into EU CR0.1[13] (note the bit # difference). See Exceptions and ISA Execution Environment. This field is ignored if DS Function Enable is DISABLED.
	6:0	**Reserved** Format: MBZ
4..5	63:10	**Scratch Space Base Pointer** Format: GeneralStateOffset[63:10]ScratchSpace Specifies the starting location of the scratch space area allocated to this FF unit as a 1K-byte aligned offset from the General State Base Address. If required, each thread spawned by this FF unit will be allocated some portion of this space, as specified by Per-Thread Scratch Space. The computed offset of the thread-specific portion will be passed in the thread payload as Scratch Space Offset. The thread is expected to utilize "stateless" DataPort read/write requests to access scratch space, where the DataPort will cause the General State Base Address to be added to the offset passed in the request header. This field is ignored if DS Function Enable is DISABLED.
	9:4	**Reserved** Format: MBZ
	3:0	**Per-Thread Scratch Space** Format: U4 power of 2 Bytes over 1K Bytes Specifies the amount of scratch space to be allocated to each thread spawned by this FF unit.The driver must allocate enough contiguous scratch space, starting at the Scratch Space Base Pointer, to ensure that the Maximum Number of Threads can each get Per-Thread Scratch Space size without exceeding the driver-allocated scratch space. This field is ignored if DS Function Enable is DISABLED. **Value** / **Name** [0,11] indicating [1K Bytes, 2M Bytes] **Programming Notes** This amount is available to the kernel for information only. It will be passed verbatim (if not altered by the kernel) to the Data Port in any scratch space access messages, but the Data Port will ignore it.
6	31:25	**Reserved** Format: MBZ

3DSTATE_DS

24:20 Dispatch GRF Start Register For URB Data

Format:	GRFRegister[4:0]

Specifies the starting GRF register number for the URB portion (Constant + Vertices) of the thread payload. This field is ignored if DS Function Enable is DISABLED.
When SIMD8_SINGLE_OR_DUAL_PATCH dispatch mode is selected, HW shall increment the GRF start register by 1 when a dual patch simd8 thread is dispatched.

Value	Name	Description
[0,31]		indicating GRF [R0, R31]

Programming Notes
When SIMD8_SINGLE_OR_DUAL_PATCH dispatch mode is selected, SW shall program this field to a value of 6 or greater to accommodate either the DUAL_PATCH or SIMD8_SINGLE_PATCH payloads.

19:18 Reserved

Format:	MBZ

17:11 Patch URB Entry Read Length

Format:	U7

Specifies how much data (in 256-bit units) is to be read from the Patch URB entry and passed in the DS thread payload.
This field is ignored if DS Function Enable is DISABLED.

Value	Name
[0,64]	

10 Reserved

Format:	MBZ

9:4 Patch URB Entry Read Offset

Format:	U6

Specifies the offset (in 256-bit units) at which Patch URB data is to be read from the URB before being included in the thread payload.
This field is ignored if DS Function Enable is DISABLED.

Value	Name
[0,63]	

3:0 Reserved

Format:	MBZ

7 31 Reserved

Format:	MBZ

30 Reserved

Format:	MBZ

3DSTATE_DS

29:21 Maximum Number of Threads

Format:	U9-1 Thread Count

Specifies the maximum number of simultaneous DS threads allowed to be active. Used to avoid using up the scratch space. Programming the value of the max threads over the number of threads based off number of threads supported in the execution units may improve performance since the architecture allows threads to be buffered between the check for max threads and the actual dispatch into the EU. Programming the max values to a number less than the number of threads supported in the execution units may reduce performance.
This field is ignored if DS Function Enable is DISABLED.

Value	Name	Description
[0,111]		indicating thread count of [1,112]

20:11 Reserved

Format:	MBZ

10 Statistics Enable

Format:	Enable

If ENABLED, this FF unit will engage in statistics gathering. Refer to the Statistics Gathering section.
If DISABLED, statistics information associated with this FF stage will be left unchanged.
This field is ignored if DS Function Enable is DISABLED.

9:5 Reserved

Format:	MBZ

4:3 Dispatch Mode

Format:	U2

This field specifies how the DS stage generates DS thread requests, and correspondingly impacts the DS thread payload. The setting of this field must agree with how the DS kernel was compiled.
This field is ignored if DS Function Enable is DISABLED.

Value	Name	Description	Programming Notes
0h	SIMD4X2	DS threads are passed one patch, up to 2 domain point inputs, and up to two output vertex handles. The DS kernel (at KSP) is expected to run in SIMD4x2 execution mode. The DUAL_PATCH KSP is ignored. The **Single Domain Point Dispatch** field can be used to force single domain point dispatches.	

3DSTATE_DS

		1h	SIMD8_SINGLE_PATCH	DS threads are passed one patch, up to 8 domain point inputs, and up to 8 output vertex handles. All the domain points are associated with the single input patch. The DS kernel (at KSP) is expected to run in SIMD8 execution mode. The DUAL_PATCH KSP is ignored.	
		2h	SIMD8_SINGLE_OR_DUAL_PATCH	SIMD8_SINGLE_OR_DUAL_PATCH This mode enables use of both the KSP and the DUAL_PATCH KSP. The KSP kernel operates just like in SIMD8_SINGLE_PATCH mode. See DUAL_PATCH Thread Execution for a discussion of how the DUAL_PATCH KSP is used.	At least 2 HS URB handles must be allocated in order to enable this mode. SIMD8_SINGLE_OR_DUAL_PATCH must not be used if the domain shader kernel uses primitive id.
		3h	Reserved		
	2	**Compute W Coordinate Enable** Format: Enable If ENABLED, the DS unit will (for each domain point) compute W = 1 - (U + V) and pass the result as a floating point value in the DS thread payload. If DISABLED, 0.0 will be passed. This field must only be ENABLED for the tessellation of TRI domains, where UVW coordinates are required. This field must be DISABLED for other domains (as they only require UV coordinates) otherwise the computed W coordinate is UNDEFINED. This field is ignored if DS Function Enable is DISABLED.			
	1	**Cache Disable** Format: Disable This bit controls the operation of the DS Cache. This field is ignored if DS Function Enable is DISABLED. If the DS Cache is DISABLED and the DS Function is ENABLED, the DS Cache is not used and all incoming domain points will be passed to DS threads. If the DS Cache is ENABLED and the DS Function is ENABLED, incoming domain points that do not hit in the DS Cache will be passed to DS threads. The DS Cache is invalidated whenever the DS Cache becomes DISABLED , whenever the DS Function Enable toggles, and between patches.			

3DSTATE_DS

	0	**Function Enable** Format: Enable If ENABLED, DS threads will be spawned to process incoming domain points which miss in the DS cache. If DISABLED, the DS stage goes into pass-through mode and performs no specific processing. This field is always used. **Programming Notes** The tessellation stages (HS, TE and DS) must be enabled/disabled as a group. I.e., draw commands can only be issued if all three stages are enabled or all three stages are disabled, otherwise the behavior is UNDEFINED.
8	31:27	**Reserved** Format: MBZ
	26:21	**Vertex URB Entry Output Read Offset** Format: U6 Specifies the offset (in 256-bit units) at which Vertex URB data is to be read from the URB by SBE. **Value** | **Name** [0,63] |
	20:16	**Vertex URB Entry Output Length** Format: U5 Specifies the amount of URB data written for each Vertex URB entry, in 256-bit register increments. **Value** | **Name** [1,16] | **Programming Notes** This length does not include the vertex header.
	15:8	**User Clip Distance Clip Test Enable Bitmask** Format: U8 This 8 bit mask field selects which of the 8 user clip distances against which trivial reject / trivial accept / must clip determination needs to be made. DX10 allows simultaneous use of ClipDistance and Cull Distance test of up to 8 distances.
	7:0	**User Clip Distance Cull Test Enable Bitmask** Format: U8 This 8 bit mask field selects which of the 8 user clip distances against which trivial reject / trivial accept determination needs to be made (does not cause a must clip). DX10 allows simultaneous use of ClipDistance and Cull Distance test of up to 8 distances.

3DSTATE_DS		
9..10	63:6	**DUAL_PATCH Kernel Start Pointer** Format: InstructionBaseOffset[63:6]Kernel This field specifies the starting location of the DUAL_PATCH kernel program run by threads spawned by this FF unit. It is specified as a 64-byte-granular offset from the Instruction Base Address. This field is ignored if DS Function Enable is DISABLED. See DUAL_PATCH Thread Execution for a discussion of how the DUAL_PATCH KSP is used.
	5:0	**Reserved** Format: MBZ

3DSTATE_GS

3DSTATE_GS

Source: RenderCS
Length Bias: 2

Controls the GS stage hardware.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 11h 3DSTATE_GS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Value: 8h — Name: Excludes DWord (0,1) **[Default]**
1..2	63:6	**Kernel Start Pointer** Format: InstructionBaseOffset[63:6]Kernel This field specifies the starting location (1st GEN4 core instruction) of the kernel program run by threads spawned by this FF unit. It is specified as a 64-byte-granular offset from the Instruction Base Address.
	5:0	**Reserved** Format: MBZ

3DSTATE_GS

3 / 31 — Single Program Flow

Specifies the initial condition of the kernel program as either a single program flow (SIMDnxm with m = 1) or as multiple program flows (SIMDnxm with m > 1). See CR0 description in ISA Execution Environment.

Value	Name	Description
0h	Disable	Single Program Flow disabled
1h	Enable	Single Program Flow enabled

30 — Vector Mask Enable

Format:	Enable Enumerated Type

Upon subsequent GS thread dispatches, this bit is loaded into the EU's Vector Mask Enable (VME, cr0.0[3]) thread state. Refer to EU documentation for the definition and use of VME state.

Value	Name	Description
0h	Dmask	The EU will use the Dispatch Mask (supplied by the GS stage) for instruction execution.
1h	Vmask	The EU will use the Vector Mask (derived from Dispatch Mask) for instruction execution.

Programming Notes
Under normal conditions SW shall specify DMask, as the GS stage will provide a Dispatch Mask appropriate to SIMD4x2 or SIMD8 thread execution (as a function of dispatch mode). E.g., for SIMD4x2 execution, the GS stage will generate a Dispatch Mask that is equal to what the EU would use as the Vector Mask. For SIMD8 execution there is no known usage model for use of Vector Mask (as there is for PS shaders).

29:27 — Sampler Count

Format:	U3

Specifies how many samplers (in multiples of 4) the geometry shader kernel uses. Used only for prefetching the associated sampler state entries.

Value	Name	Description
0h	No Samplers	No Samplers used
1h	1-4 Samplers	Between 1 and 4 samplers used
2h	5-8 Samplers	Between 5 and 8 samplers used
3h	9-12 Samplers	Between 9 and 12 samplers used
4h	13-16 Samplers	Between 13 and 16 samplers used
5h-7h	Reserved	

26 — Reserved

3DSTATE_GS

25:18 Binding Table Entry Count

Format:	U8

When **HW Generated Binding Table** is disabled:
Specifies how many binding table entries the kernel uses. Used only for prefetching of the binding table entries and associated surface state.
Note: For kernels using a large number of binding table entries, it may be wise to set this field to zero to avoid prefetching too many entries and thrashing the state cache.
When **HW Generated Binding Table** bit is enabled:
This field indicates which cache lines (512bit units - 32 Binding Table Entry section) should be fetched. Each bit in this field corresponds to a cache line. Only the 1st 4 non-zero Binding Table entries of each 32 Binding Table entry section prefetched will have its surface state prefetched.

Programming Notes
When HW binding table bit is set, it is assumed that the Binding Table Entry Count field will be generated at JIT time.

17 Thread Dispatch Priority

Specifies the priority of the thread for dispatch.

Value	Name	Description
0h	Normal	Normal thread dispatch priority
1h	High	High thread dispatch priority

16 Floating Point Mode

Specifies the initial floating point mode used by the dispatched thread.

Value	Name	Description
0h	IEEE-754	Use IEEE-754 Rules
1h	Alternate	Use alternate rules

15:14 Reserved

Format:	MBZ

13 Illegal Opcode Exception Enable

Format:	Enable

This bit gets loaded into EU CR0.1[12] (note the bit # difference). See *Exceptions and ISA Execution Environment*.

12 Accesses UAV

Format:	Enable

This field must be set when GS has a UAV access.

Programming Notes
This field must not be set when GS Function Enable is disabled.

11 Mask Stack Exception Enable

Format:	Enable

This bit gets loaded into EU CR0.1[11]. See *Exceptions and ISA Execution Environment.*

3DSTATE_GS

	10:8	**Reserved** Format: MBZ
	7	**Software Exception Enable** Format: Enable This bit gets loaded into EU CR0.1[13] (note the bit # difference). See *Exceptions and ISA Execution Environment*.
	6	**Reserved** Format: MBZ
	5:0	**Expected Vertex Count** Format: U6 Specifies the number of vertices per input object expected by the GS thread. Input topologies not matching this expect value are discarded. Note that **DiscardAdjacency** is also considered (e.g., if the value programmed is 3 and DiscardAdjacency is set, TRILIST_ADJ and TRISTRIP_ADJ topologies are not discarded as they will pass 3 vertices/object to the GS threads). **Value** | **Name** [1,32] |
4..5	63:10	**Scratch Space Base Pointer** Format: GeneralStateOffset[63:10]ScratchSpace Specifies the starting location of the scratch space area allocated to this FF unit as a 1K-byte aligned offset from the General State Base Address. If required, each thread spawned by this FF unit will be allocated some portion of this space, as specified by Per-Thread Scratch Space. The computed offset of the thread-specific portion will be passed in the thread payload as Scratch Space Offset. The thread is expected to utilize "stateless" DataPort read/write requests to access scratch space, where the DataPort will cause the General State Base Address to be added to the offset passed in the request header. This field is ignored if VS Function Enable is DISABLED.
	9:4	**Reserved** Format: MBZ
	3:0	**Per-Thread Scratch Space** Format: U4 power of 2 Bytes over 1K Bytes Specifies the amount of scratch space to be allocated to each thread spawned by this FF unit. The driver must allocate enough contiguous scratch space, starting at the Scratch Space Base Pointer, to ensure that the Maximum Number of Threads can each get Per-Thread Scratch Space size without exceeding the driver-allocated scratch space. **Value** | **Name** | **Description** [0,11] | | indicating [1K Bytes, 2M Bytes]
6	31	**Reserved** Format: MBZ

3DSTATE_GS

30:29 Dispatch GRF Start Register For URB Data [5:4]

Format:	U2

Specifies bit [5:4] of the starting GRF register number for the URB portion (Constant + Vertices) of the thread payload. The **Dispatch GRF Start Register For URB Data [3:0]** field is used to specify bits [3:0] of the starting GRF register number.

28:23 Output Vertex Size

Format:	U6

[0,63] indicating [1,64] 16B units
Specifies the size of each vertex stored in the GS output entry (following any Control Header data) as a number of 128-bit units (minus one).

Programming Notes

Programming Restrictions: The vertex size must be programmed as a multiple of 32B units with the following exception: Rendering is disabled (as per SOL stage state) and the vertex size output by the GS thread is 16B.

If rendering is enabled (as per SOL state) the vertex size must be programmed as a multiple of 32B units. In other words, the only time software can program a vertex size with an odd number of 16B units is when rendering is disabled.

22:17 Output Topology

Format:	3D_Prim_Topo_Type

This field specifies the topology type (3DPrimType) to be associated with GS-thread output vertices (if any).

16:11 Vertex URB Entry Read Length

Specifies the amount of URB data read and passed in the thread payload for each Vertex URB entry, in 256-bit register increments.

Programming Notes

Programming Restriction:This field must be a non-zero value if Include Vertex Handles is cleared to zero.

10 Include Vertex Handles

Format:	Boolean

If set, all the input Vertex URB handles are included in the payload. These are referred to as "pull model" URB handles, as the thread will use them to read from the URB.

Programming Notes

Programming Restriction: This field must be set if Vertex URB Entry Read Length is cleared to zero.

3DSTATE_GS

9:4 Vertex URB Entry Read Offset

Format:	U6

Specifies the offset (in 256-bit units) at which Vertex URB data is to be read from the URB before being included in the thread payload. This offset applies to all Vertex URB entries passed to the thread.

3:0 Dispatch GRF Start Register For URB Data

Format:	U4

Specifies the starting GRF register number for the URB portion (Constant + Vertices) of the thread payload.

Value	Name	Description
[0,15]		indicating GRF [R0, R15]

Programming Notes

If Include Vertex Handles is enabled (pull or hybrid handles case), then
For simd4x2:
For DUAL_OBJECT dispatch mode this field should be:
(((2*numVerticesPerObject) + 8 - 1)/8) + 1
For SINGLE and DUAL_INSTANCE dispatch modes this field should be:
((numVerticesPerObject +8 - 1)/8) + 1
If Include Primitive ID is set, then add 1 to the value obtained by using the above

If Include Vertex Handles is enabled (pull or hybrid handles case), then
For SKL simd8:
For InstanceCount == 1:
numVerticesPerObject 2
For InstanceCount > 1:
((numVerticesPerObject 8 - 1)/8) 2
If Include Primitive ID is set, then add 1 to the value obtained by using the above

When Include Vertex Handles is set for non-instanced SIMD8 dispatch of PATCHLIST_14..32 objects, pushed vertex data and/or pushed constants cannot be used as they would need to start in the payload beyond the range of this field (i.e., beyond R15). When Include PrimitiveID is also set, this issue extends to non-instanced SIMD8 dispatch of PATCHLIST_13..32 objects.

7 31:26 Reserved

Format:	MBZ

25:24 Reserved

Format:	MBZ

23:20 Control Data Header Size

Format:	U4

Specifies the number of 32B units of control data header located at the start of the GS URB entry. The value 0 indicates there is no control data header, and Control Data Format is ignored. Software must ensure that the Control Data Header Size is sufficient to accommodate the maximum number of vertices output by the GS thread. It is UNDEFINED for a GS thread to report

3DSTATE_GS

more output vertices than can be accommodated in a non-zero-sized header. (If the header size is zero, by definition neither cut nor StreamID bits are defined.

Value	Name
[0,8]	32B Units

19:15 **Instance Control**

Format:	U5-1 #Instances

Specifies the number of instances (minus one) for each input object. To avoid confusion, this document uses the term **"InstanceCount"** to refer to InstanceControl+1, with a range of [1,32]
If **InstanceCount>1**, DUAL_OBJECT mode is invalid. Software will likely want to use DUAL_INSTANCE mode for higher performance, but SINGLE mode is also supported.
When **InstanceCount=1** (one instance per object), software can decide which dispatch mode to use. DUAL_OBJECT mode would likely be the best choice for performance, followed by SINGLE mode. DUAL_INSTANCE mode is not recommended but is supported.

Value	Name	Description
[0,31]		Indicating [1,31] instances

14:13 **Default Stream Id**

Format:	U2

When the GS is enabled, unless the GS output entry contains StreamID bits in the control header, this field specifies the default StreamID associated with any GS-thread output vertices. When the GS is disabled, StreamID will be output as 0.

12:11 **Dispatch Mode**

Format:	U2

This field specifies how the GS unit dispatches multiple instances and/or multiple objects.

Value	Name	Description	Programming Notes
0h	Single	Each thread shades a single instance of one object.	
1h	Dual Instance	Each thread shades possibly two instances of one object. If the InstanceCount is odd, a trailing dispatch of only one instance will be made for each object received. Not recommended if InstanceCount = 1, assuming a kernel optimized for SINGLE or DUAL_OBJECT dispatch would outperform a kernel compiled for DUAL_INSTANCE but only passed one instance.	
2h	Dual Object	Each thread shades one instance of possibly two objects. The GS unit attempt to pair objects together into one dispatch, but under some circumstances only one object may be dispatched (as controlled by the DispatchMask generated by the GS unit). Not valid for objects	

3DSTATE_GS

		with more than 16 vertices per object. Not valid if InstanceCount > 1 (more than one instance per object).	
3h	SIMD8	Each thread shades up to 8 different objects or (if InstanceCount >1) 8 instances of a single object.	The driver must send pipe control with a cs stall after a 3dstate_gs state change and the Dispatch Mode is simd8 and the number of handles allocated to gs is less than 16.

Programming Notes
The GS must be allocated at least two URB handles or behavior is UNDEFINED for Dual Instance or Dual Object mode.

10 **Statistics Enable**

Format:	Enable

This bit controls whether GS-unit-specific statistics register(s) can be incremented.

Value	Name	Description
0h	Disable	GS_INVOCATIONS_COUNT and GS_PRIMITIVES_COUNT cannot increment
1h	Enable	GS_INVOCATIONS_COUNT and GS_PRIMITIVES_COUNT can increment

9:5 **Invocations Increment Value**

Format:	U5

Specifies how much to increment the GS_INVOCATIONS_COUNT for each instance of each object. This control is provided to allow software to process multiple instances (from an API POV) in a single kernel invocation.

In SINGLE dispatch mode, the counter will increment by this value for each dispatch (as it's only one instance of one object).

In DUAL_INSTANCE mode, the counter will be incremented by the value if only one instance is included in the dispatch (i.e., the last odd instance), otherwise the counter will be incremented by twice this value.

In DUAL_OBJECT dispatch mode, the counter will be incremented by the value if only one object is included in the dispatch (i.e., a forced dispatch of one object), otherwise the counter will be incremented by twice this value.

Value	Name	Description
[0,31]		indicating an increment of [1,32]

4 **Include Primitive ID**

Format:	Boolean

If set, R1 of the payload is written with Primitive ID value(s).
If clear, these Primitive ID values are not included in the payload R1.

3DSTATE_GS

3 — Hint

Format:	U1

This state bit is simply passed in GS thread payloads for use by the GS kernel - it has no other impact on hardware operation.

2 — Reorder Mode

This bit controls how vertices of triangle objects resulting from TRISTRIP[_ADJ][_REV] topologies are [re]ordered when passed in the GS thread payload See Object Vertex Ordering table (below).

Value	Name	Description
0h	LEADING	Reorder the vertices of alternating triangles of a TRISTRIP[_ADJ] such that the leading (first) vertices are in consecutive order starting at v0. A similar reordering is performed on alternating triangles in a TRISTRIP_REV.
1h	TRAILING	Reorder the vertices of alternating triangles of a TRISTRIP[_ADJ] such that the trailing (last) vertices are in consecutive order starting at v2. A similar reordering is performed on alternating triangles in a TRISTRIP_REV.

1 — Discard Adjacency

Format:	Enable

When set, adjacent vertices will not be passed in the GS payload when objects with adjacency are processed. Instead, only the non-adjacent vertices will be passed in the same fashion as the without-adjacency form of the primitive. Software should set this bit whenever a GS kernel is used that does not expect adjacent vertices. This allows both with-adjacency/without-adjacency variants of the primitive to be submitted to the pipeline (via 3DPRIMITIVE) - the GS unit will silently discard any adjacent vertices and present the GS thread with only the internal object. When clear, adjacent vertices will be passed to the GS thread, as dictated by the incoming primitive type. Software should only clear this bit when a GS kernel is used that does expect adjacent vertices. E.g., if the GS kernel is compiled to expect a TRIANGLE_ADJ object, software must clear this bit. Software should also clear this bit if the GS kernel expects a POINT or PATCHLIST_n object (which don't have with-adjacency variants).

The only hardware assistance is to allow the submission of a with-adjacency variant of a primitive when operating with a GS kernel that expects the without-adjacency variant of the object. (E.g., when the GS kernel is compiled to expect a TRIANGLE object, software should set this bit just in case a TRILIST_ADJ is submitted to the pipeline.) Note that the GS unit is otherwise not aware of the object type that is expected by the GS kernel. It is up to software to ensure that the submitted primitive type (in 3DPRIMITIVE) is otherwise compatible with the object type expected by the GS kernel. (E.g., if the GS kernel expects a LINE_ADJ object, only LINELIST_ADJ or LINESTRIP_ADJ should be submitted, otherwise the GS kernel will produce unpredictable results.) Also note that it is possible to craft a GS kernel which can accept any object type that's thrown at it by first examining the PrimType passed in the payload and then using this info to correctly interpret the number of vertices passed in the payload.

0 — Enable

Format:	Enable

Specifies whether the GS stage is enabled or disabled (pass-through).

3DSTATE_GS

8 — 31 — Control Data Format

Format:	U1

This field specifies the format of the control data header (if any).

Value	Name	Description
0h	CUT	The control data header contains cut bits.
1h	SID	The control data header contains StreamID bits. . Output Topology must be set to POINTLIST, or behavior is UNDEFINED.

30 — Static Output

Format:	Enable

Specifies whether the GS shader outputs a static number of vertices per invocation.
If this bit is clear, the number of vertices output by each GS shader invocation is stored by the GS thread at the very beginning of the output URB entry (see GS URB Entry section below).

29:27 — Reserved

Format:	MBZ

26:16 — Static Output Vertex Count

Format:	U11 Count of object vertices

If GSEnable is set and StaticOutput is set, this field specifies the total number of vertices output each GS shader invocation.
If **GSEnable** is set and StaticOutput is clear (variable GS output), the total number of vertices output by a GS shader invocation is stored by the thread at the very beginning of the output URB entry. This field is then ignored. (See GS URB Entry below).

15:9 — Reserved

Format:	MBZ

8:0 — Maximum Number of Threads

Format:	U9-1 Thread count

Specifies the maximum number of simultaneous threads allowed to be active. Used to avoid using up the scratch space. Programming the value of the max threads over the number of threads based off number of threads supported in the execution units may improve performance since the architecture allows threads to be buffered between the check for max threads and the actual dispatch into the EU. Programming the max values to a number less than the number of threads supported in the execution units may reduce performance.

Value	Name	Description
[7,111]		Indicating thread count of [8,112]

9 — 31:27 — Reserved

Format:	MBZ

26:21 — Vertex URB Entry Output Read Offset

Format:	U6

Specifies the offset (in 256-bit units) at which Vertex URB data is to be read from the URB by SBE.

3DSTATE_GS

		Value / **Name** [0,63]
	20:16	**Vertex URB Entry Output Length** Format: U5 Specifies the amount of URB data written for each Vertex URB entry, in 256-bit register increments. **Value** / **Name** [1,16] **Programming Notes** This length does not include the vertex header.
	15:8	**User Clip Distance Clip Test Enable Bitmask** Format: Enable[8] This 8 bit mask field selects which of the 8 user clip distances against which trivial reject / trivial accept / must clip determination needs to be made. DX10 allows simultaneous use of ClipDistance and Cull Distance test of up to 8 distances.
	7:0	**User Clip Distance Cull Test Enable Bitmask** Format: Enable[8] This 8 bit mask field selects which of the 8 user clip distances against which trivial reject / trivial accept determination needs to be made (does not cause a must clip). DX10 allows simultaneous use of ClipDistance and Cull Distance test of up to 8 distances.

3DSTATE_HS

3DSTATE_HS

Source: RenderCS

Length Bias: 2

Controls the HS stage hardware.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 1Bh 3DSTATE_HS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Value: 7 — Name: Excludes DWord (0,1) **[Default]**
1	31:30	**Reserved** Format: MBZ
	29:27	**Sampler Count** Format: U3 Specifies how many samplers (in multiples of 4) the HS kernels use. Used only for prefetching the associated sampler state entries. 0h — No Samplers — no samplers used 1h — 1-4 Samplers — between 1 and 4 samplers used 2h — 5-8 Samplers — between 5 and 8 samplers used

3DSTATE_HS

3h	9-12 Samplers	between 9 and 12 samplers used
4h	13-16 Samplers	between 13 and 16 samplers used
5h-7h	Reserved	Reserved

26 Reserved

Format:	MBZ

25:18 Binding Table Entry Count

Format:	U8

When HW Generated Binding Table is disabled:
Specifies how many binding table entries the kernel uses. Used only for prefetching of the binding table entries and associated surface state.
Note: For kernels using a large number of binding table entries, it may be wise to set this field to zero to avoid prefetching too many entries and thrashing the state cache.

Programming Notes
When HW binding table bit is set, it is assumed that the Binding Table Entry Count field will be generated at JIT time.

17 Thread Dispatch Priority
Specifies the priority of the thread for dispatch

Value	Name	Description
0h	Normal	Normal Priority
1h	High	High Priority

16 Floating Point Mode
Specifies the initial floating point mode used by the dispatched thread.

Value	Name	Description
0h	IEEE-754	Use IEEE-754 Rules
1h	alternate	Use alternate rules

15:14 Reserved

Format:	MBZ

13 Illegal Opcode Exception Enable

Format:	Enable

This bit gets loaded into EU CR0.1[12] (note the bit # difference). See Exceptions and ISA Execution Environment.

12 Software Exception Enable

Format:	Enable

This bit gets loaded into EU CRO1[13] (note the bit # difference). See Exceptions and ISA Execution Environment.

11:8 Reserved

Format:	MBZ

7:0 Reserved

3DSTATE_HS

		Format: MBZ
2	31	**Enable** Format: Enable Specifies whether the HS function is enabled or disabled (pass-through). If ENABLED MI_TOPOLOGY_FILTER must be used to silently discard any topologies that the HS kernel is not expecting. E.g., if the HS kernel is expecting PATCHLIST_32 topologies, MI_TOPOLOGY_FILTER must be set to PATCHLIST_32 so only those topologies can reach the enabled HS. **Programming Notes** The tessellation stages (HS, TE and DS) must be enabled/disabled as a group. I.e., draw commands can only be issued if all three stages are enabled or all three stages are disabled, otherwise the behavior is UNDEFINED.
	30	**Reserved** Format: MBZ
	29	**Statistics Enable** Format: Enable This bit controls whether HS-unit-specific statistics register(s) will increment (for each patch).
	28:27	**Reserved** Format: MBZ
	26:18	**Reserved** Format: MBZ
	17	**Reserved** Format: MBZ
	16:8	**Maximum Number of Threads** Format: U9-1 Specifies the maximum number of simultaneous threads allowed to be active. Used to avoid using up the scratch space. Programming the value of the max threads over the number of threads based off number of threads supported in the execution units may improve performance since the architecture allows threads to be buffered between the check for max threads and the actual dispatch into the EU. Programming the max values to a number less than the number of threads supported in the execution units may reduce performance. **Value** / **Name** / **Description** [0,335] / / indicating thread count of [1,336]
	7:5	**Reserved** Format: MBZ
	4	**Reserved** Format: MBZ
	3:0	**Instance Count** Format: U4-1

3DSTATE_HS

This field determines the number of threads (minus one) spawned per input patch.
If the HS kernel uses a barrier function, software must restrict the **Instance Count** to the number of threads that can be simultaneously active within a subslice. Factors which must be considered includes scratch memory availability.

Value	Name	Description
[0,15]		representing [1,16] instances

Programming Notes
Instance Count must be programmed to 0 (1 instance) whenever DispatchMode is programmed to DUAL_PATCH.
A pipe_control with cs stall must be sent whenever the HS_STATE.InstanceCount changes from 0 (no instancing) to >0 (instancing) or when there is transition from HS_STATE.Enabled = false to (HS_STATE.Enabled = true && InstanceCount > 0).

3..4	63:6	**Kernel Start Pointer** Format: InstructionBaseOffset[63:6]Kernel This field specifies the starting location (1st GEN core instruction) of the kernel program run by threads spawned by this FF unit. It is specified as a 64-byte-granular offset from the Instruction Base Address.
	5:0	**Reserved** Format: MBZ
5..6	63:10	**Scratch Space Base Pointer** Format: GeneralStateOffset[63:10] Value / Name / Description: [0,31] / / Specifies the location of the scratch space area allocated to this FF unit, specified as a 1KB-granular offset from the General State Base Address. If required, each thread spawned by this FF unit will be allocated some portion of this space, as specified by Per-Thread Scratch Space.
	9:4	**Reserved** Format: MBZ
	3:0	**Per-Thread Scratch Space** Format: U4 power of 2 Bytes over 1K Bytes Specifies the amount of scratch space to be allocated to each thread spawned by this FF unit. The driver must allocate enough contiguous scratch space, starting at the Scratch Space Base Pointer, to ensure that the Maximum Number of Threads can each get Per-Thread Scratch Space size without exceeding the driver-allocated scratch space. Value / Name / Description: [0,11] / / Indicating[1K Bytes, 2M Bytes
7	31:29	**Reserved**

3DSTATE_HS

	Format:	MBZ

28 Dispatch GRF Start Register For URB Data [5]

Format:	U1

Specifies bit [5] of the starting GRF register number for the URB portion (Constant + Vertices) of the thread payload. The **Dispatch GRF Start Register For URB Data [4:0]** field is used to specify bits [4:0] of the starting GRF register number.

27 Single Program Flow

Format:	Enable

Specifies the initial condition of the kernel program as either a single program flow (SIMDnxm with m = 1) or as multiple program flows (SIMDnxm with m > 1). See CR0 description in *ISA Execution Environment*.

Value	Name	Description
0h	Reserved	
1h	Enable	Single Program Flow Enabled

26 Vector Mask Enable

Format:	U1 Enumerated Type

Upon subsequent HS thread dispatches, this bit is loaded into the EU's Vector Mask Enable (VME, cr0.0[3]) thread state. Refer to the EU documentation for the definition and use of VME state.

Value	Name	Description
0h	Dmask	The EU will use the Dispatch Mask (supplied by the HS stage) for instruction execution.
1h	Vmask	The EU will use the Vector Mask (derived from the Dispatch Mask) for instruction execution.

Programming Notes
Under normal conditions SW shall specify DMask, as the HS stage will provide a Dispatch Mask appropriate to SIMD4x2 or SIMD8 thread execution (as a function of dispatch mode). E.g., for SIMD4x2 thread execution, the HS state will generate a Dispatch Mask that is equal to what the EU would use as a Vector Mask. For SIMD8 execution there is no known usage model for use of Vector Mask (as there is for PS shaders).

25 Accesses UAV

Format:	Enable

This field must be set when HS has a UAV access

Programming Notes
This field must not be set when HS Function Enable is disabled.

24 Include Vertex Handles

Format:	Boolean

If set, all the input Vertex URB handles are included in payloads.

3DSTATE_HS

This field is ignored if **HS Function Enable** is DISABLED.

Programming Notes
Programming Restriction: This field must be set if value if **Vertex URB Entry Read Length** is cleared to zero.

23:19 **Dispatch GRF Start Register For URB Data**

Format:	U5

Specifies the starting GRF register number for the URB portion (Constant + Vertices) of the thread payload.

This field is ignored if **HS Function Enable** is DISABLED.

Value	Name	Description
[0,31]		indicating GRF [R0, R31]

Programming Notes
When Include Vertex Handles is set for non-instanced 8_PATCH dispatch of PATCHLIST_30..32 objects, pushed vertex data and/or pushed constants cannot be used as they would need to start in the payload beyond the range of this field (i.e., beyond R31). When Include PrimitiveID is also set, this issue extends to non-instanced 8_PATCH dispatch of PATCHLIST_29..32 objects.

18:17 **Dispatch Mode**

Format:	U2

This field is unused to set the current thread dispatch mode for the HS stage.

Value	Name	Description
0h	SINGLE_PATCH	HS threads are passed inputs and an output handle associated with a single input patch.
1h	DUAL_PATCH	HS threads are passed inputs and an output handle associated with (up to) two input patches. Patch 0 data is passed in the four lower channels while Patch 1 data (if present) is passed in the four upper channels. Restrictions: Only valid for 4 or fewer input control points (PATCHLIST_4 and below). SW is expected to use MI_TOPOLOGY_FILTER to ensure only patches with the expected # of ICPs are processed by the pipeline when HS is enabled.
2h	8_PATCH	HS threads are passed inputs and an output handle associated with (up to) 8 patches in SIMD8 fashion. See 8_PATCH Thread Payload.
3h	Reserved	

3DSTATE_HS

	16:11	**Vertex URB Entry Read Length** Format: U6 Specifies the amount of URB data read and passed in the thread payload for each Vertex URB entry, in 256-bit register increments. This field is ignored if HS Function Enable is DISABLED. **Value** [0,63] — **Name** **Programming Notes** **Programming Restriction:** This field must be a non-zero value if **Include Vertex Handles** is cleared to zero.
	10	**Reserved** Format: MBZ
	9:4	**Vertex URB Entry Read Offset** Format: U6 Specifies the offset (in 256-bit units) at which Vertex URB data is to be read from the URB before being included in the thread payload. This offset applies to all Vertex URB entries passed to the thread. This field is ignored if HS Function Enable is DISABLED. **Value** [0,63] — **Name**
	3:1	**Reserved** Format: MBZ
	0	**Include Primitive ID** Format: Enable If set, R1 of the payload is written with Primitive ID value(s). If clear, these Primitive IDs are not included in the payload R1. **Programming Notes** This field is only used when DUAL_PATCH DispatchMode is specified. In SINGLE_PATCH, the single Primitive ID is always passed in R0.
8	31:0	**Reserved** Format: MBZ

3DSTATE_URB_DS

3DSTATE_URB_DS

Source: RenderCS

Length Bias: 2

Description

This command may not overlap with the push constants in the URB defined by the 3DSTATE_PUSH_CONSTANT_ALLOC_VS, 3DSTATE_PUSH_CONSTANT_ALLOC_DS, 3DSTATE_PUSH_CONSTANT_ALLOC_HS, and 3DSTATE_PUSH_CONSTANT_ALLOC_GS commands.

The URB Starting Address and Number of URB Entries fields shall be programmed as if there is only one slice enabled. When more than one slice is enabled, hardware will (a) recompute the actual URB Starting Address based on the number of enabled slices and (b) multiply the Number of URB Entries by the number of enabled slices. Software shall ensure that the values programmed do not exceed the URB capacity of a single slice. Refer to the L3 allocation and programming guide for valid URB configurations.

Programming Notes

When programming DS URB state for the RCS 3D pipe, 3DSTATE_URB_VS, 3DSTATE_URB_HS, and 3DSTATE_URB_GS must also be programmed in order for the programming of this state to be valid.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 32h 3DSTATE_URB_DS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n

3DSTATE_URB_DS

1 — 31:25 — DS URB Starting Address

Format:	U7

Offset from the start of the URB memory where DS starts its allocation, specified in multiples of 8 KB.

Value	Name	Exists If
[0,48]		Device[SliceCount] == 1
[4,48]		Device[SliceCount] GT 1

24:16 — DS URB Entry Allocation Size

Format:	U9-1 Count of 512-bit units

Specifies the length of each URB entry owned by DS. This field is always used (even if DS Function Enable is DISABLED).

Value	Name
[0,9]	

15:0 — DS Number of URB Entries

Description
Specifies the number of URB entries that are used by DS, based on only 1 slice enabled. When multiple slices are enabled, HW will multiply the value programmed by the number of slices in order to determine the total number of entries. SW shall ensure that the total number of entries does not exceed the relevant ValidValue range listed below. This field is always used (even if DS Function Enable is DISABLED).
If Domain Shader Thread Dispatch is Enabled then the minimum number of handles that must be allocated is 34 URB entries.

Value	Name
[0,416]	

Programming Notes
DS Number of URB Entries must be divisible by 8 if the DS URB Entry Allocation Size is programmed to a value less than 9, which is 10 512-bit URB entries. "2:0" = reserved "000"

3DSTATE_URB_GS

3DSTATE_URB_GS

Source: RenderCS

Length Bias: 2

Description

This command may not overlap with the push constants in the URB defined by the 3DSTATE_PUSH_CONSTANT_ALLOC_VS, 3DSTATE_PUSH_CONSTANT_ALLOC_DS, 3DSTATE_PUSH_CONSTANT_ALLOC_HS, and 3DSTATE_PUSH_CONSTANT_ALLOC_GS commands.

The URB Starting Address and Number of URB Entries fields shall be programmed as if there is only one slice enabled. When more than one slice is enabled, hardware will (a) recompute the actual URB Starting Address based on the number of enabled slices and (b) multiply the Number of URB Entries by the number of enabled slices. Software shall ensure that the values programmed do not exceed the URB capacity of a single slice. Refer to the L3 allocation and programming guide for valid URB configurations

Programming Notes

When programming GS URB state for the RCS 3D pipe, 3DSTATE_URB_VS, 3DSTATE_URB_HS, and 3DSTATE_URB_DS must also be programmed in order for the programming of this state to be valid.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 33h 3DSTATE_URB_GS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n

3DSTATE_URB_GS

1 — 31:25 — GS URB Starting Address

Format:	U7

Offset from the start of the URB memory where GS starts its allocation, specified in multiples of 8 KB.

Value	Name	Exists If
[0,48]		Device[SliceCount] == 1
[4,48]		Device[SliceCount] GT 1

24:16 — GS URB Entry Allocation Size

Format:	U9-1 512-bit units

Specifies the length of each URB entry owned by GS. This field is always used (even if GS Function Enable is DISABLED).

15:0 — GS Number of URB Entries

Specifies the number of URB entries that are used by GS, based on only 1 slice enabled. When multiple slices are enabled, HW will multiply the value programmed by the number of slices in order to determine the total number of entries. SW shall ensure that the total number of entries does not exceed the relevant ValidValue range listed below.
This field is always used (even if GS Function Enable is DISABLED).

Value	Name
[0,256]	

Programming Notes
Only if GS is disabled can this field be programmed to 0. If GS is enabled this field shall be programmed to a value greater than 0. For GS Dispatch Mode "Single", this field shall be programmed to a value greater than or equal to 1. For other GS Dispatch Modes, refer to the definition of Dispatch Mode (3DSTATE_GS) for minimum values of this field.
GS Number of URB Entries must be divisible by 8 if the GS URB Entry Allocation Size is less than 9 512-bit URB entries. "2:0" = reserved "000"
When 3DSTATE_GS:Enable is true, the minimum number of GS Number of URB Entries must be set to 2.

3DSTATE_URB_HS

3DSTATE_URB_HS

Source: RenderCS

Length Bias: 2

Description

This command may not overlap with the push constants in the URB defined by the 3DSTATE_PUSH_CONSTANT_ALLOC_VS, 3DSTATE_PUSH_CONSTANT_ALLOC_DS, 3DSTATE_PUSH_CONSTANT_ALLOC_HS, and 3DSTATE_PUSH_CONSTANT_ALLOC_GS commands.

The URB Starting Address and Number of URB Entries fields shall be programmed as if there is only one slice enabled. When more than one slice is enabled, hardware will (a) recompute the actual URB Starting Address based on the number of enabled slices and (b) multiply the Number of URB Entries by the number of enabled slices. Software shall ensure that the values programmed do not exceed the URB capacity of a single slice. Refer to the L3 allocation and programming guide for valid URB configurations

Programming Notes

When programming HS URB state for the RCS 3D pipe, 3DSTATE_URB_VS, 3DSTATE_URB_DS, and 3DSTATE_URB_GS must also be programmed in order for the programming of this state to be valid.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 31h 3DSTATE_URB_HS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n

3DSTATE_URB_HS

1 — 31:25 — HS URB Starting Address

Format:	U7

Offset from the start of the URB memory where HS starts its allocation, specified in multiples of 8 KB.

Value	Name	Exists If
[0,48]		Device[SliceCount] == 1
[4,48]		Device[SliceCount] GT 1

24:16 — HS URB Entry Allocation Size

Format:	U9-1 Count of 512-bit units

Specifies the length of each URB entry owned by HS. This field is always used (even if HS Function Enable is DISABLED).

15:0 — HS Number of URB Entries

Specifies the number of URB entries that are used by HS, based on only 1 slice enabled. When multiple slices are enabled, HW will multiply the value programmed by the number of slices in order to determine the total number of entries. SW shall ensure that the total number of entries does not exceed the relevant ValidValue range listed below.
This field is always used (even if HS Function Enable is DISABLED).
Programming Restriction:HS Number of URB Entries must be divisible by 8 if the HS URB Entry Allocation Size is less than 9 512-bit URB entries."2:0" = reserved "000"

Value	Name
[0,256]	

Programming Notes
When 3DSTATE_HS:Enable is true and 3DSTATE_HS:Dispatch Mode is set to 8 patch, the minimum number of HS Number of URB Entries must be set to 16. When 3DSTATE_HS:Enable is true and 3DSTATE_HS:Dispatch Mode is set to dual patch, the minimum number of HS Number of URB Entries must be set to 4.

3DSTATE_URB_VS

3DSTATE_URB_VS

Source: RenderCS, PositionCS

Length Bias: 2

Description

VS URB Entry Allocation Size equal to 4(5 512-bit URB rows) may cause performance to decrease due to banking in the URB. Element sizes of 16 to 20 should be programmed with six 512-bit URB rows.

This command may not overlap with the push constants in the URB defined by the 3DSTATE_PUSH_CONSTANT_ALLOC_VS, 3DSTATE_PUSH_CONSTANT_ALLOC_DS, 3DSTATE_PUSH_CONSTANT_ALLOC_HS, and 3DSTATE_PUSH_CONSTANT_ALLOC_GS commands.

The offset and size should be programmed as if there is only one slice enabled. Hardware will grow the size based on the slice configuration. Software shall ensure that the values programmed do not exceed the URB capacity of one slice. Refer to the L3 allocation and programming guide for valid URB configurations.

Programming Notes

When programming VS URB state for the RCS 3D pipe, 3DSTATE_URB_HS, 3DSTATE_URB_DS, and 3DSTATE_URB_GS must also be programmed in order for the programming of this state to be valid.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 30h 3DSTATE_URB_VS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n

3DSTATE_URB_VS

1 — 31:25 — VS URB Starting Address

Format:	U7

Offset from the start of the URB memory where VS starts its allocation, specified in multiples of 8 KB.

Value	Name	Exists If
[0,48]		Device[SliceCount] == 1
[4,48]		Device[SliceCount] GT 1

24:16 — VS URB Entry Allocation Size

Format:	U9-1 count of 512-bit units

Specifies the length of each URB entry owned by VS. This field is always used (even if VS Function Enable is DISABLED).

Programming Notes
Programming Restriction: As the VS URB entry serves as both the per-vertex input and output of the VS shader, the VS URB Allocation Size must be sized to the maximum of the vertex input and output structures.

15:0 — VS Number of URB Entries

Format:	U16

Specifies the number of URB entries that are used by VS, based on only 1 slice enabled. When multiple slices are enabled, HW will multiply the value programmed by the number of slices in order to determine the total number of entries. SW shall ensure that the total number of entries does not exceed the relevant ValidValue range listed below.
This field is always used (even if VS Function Enable is DISABLED).

Value	Name
[34,704]	

Programming Notes
Programming Restriction: VS Number of URB Entries must be divisible by 8 if the VS URB Entry Allocation Size is less than 9 512-bit URB entries."2:0" = reserved "000b"

HCP_BSD_OBJECT

HCP_BSD_OBJECT

Source: VideoCS

Length Bias: 2

The HCP is selected with the **Media Instruction Opcode "7h"** for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.

The HCP_BSD_OBJECT command fetches the HEVC bit stream for a slice starting with the first byte in the slice. The bit stream ends with the last non-zero bit of the frame and does not include any zero-padding at the end of the bit stream. There can be multiple slices in a HEVC frame and thus this command can be issued multiple times per frame.

The HCP_BSD_OBJECT command must be the last command issued in the sequence of batch commands before the HCP starts decoding. Prior to issuing this command, it is assumed that all configuration parameters in the HCP have been loaded including workload configuration registers and configuration tables. When this command is issued, the HCP is waiting for bit stream data to be presented to the shift register.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: 20h HCP_BSD_OBJECT_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 1h; Name:

HCP_BSD_OBJECT

1	31:0	**Indirect BSD Data Length** Format: U32 Specifies the length in bytes of the bitstream data for the current slice. It includes the first byte of the slice and the last non-zero byte of the in the slice. Specifically, the zero-padding bytes (if present) and the next start-code are excluded.

Value	Name	Description
[268435456,2147483547]	Data_Length_beyond_28_bits	This is an added valid range when the bit length is extended from 28 bits to 32 bits. This is added to support 16k x 16k picture size bitstream.

2	31:29	**Reserved** Format: MBZ
	28:0	**Indirect Data Start Address** Format: U29 Specifies the byte-aligned graphics memory starting address of the slice bit stream relative to the **BSD Indirect Object Base Address**.

HCP_PIC_STATE

HCP_PIC_STATE

Source: VideoCS

Length Bias: 2

The HCP is selected with the **Media Instruction Opcode "7h"** for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.

This is a picture level command and is issued only once per workload for both encoding and decoding processes.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: 10h HCP_PIC_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 11h Name:
1	31:26	**Reserved** Format: MBZ
	25:16	**FrameHeightInMinCbMinus1** Format: U10 Specifies the height of each decoded picture in units of minimum coding block size. Value: [0-4122] Name:

HCP_PIC_STATE

		Programming Notes The decoded picture height in units of luma samples equals • (FrameHeightInMinCbMinus1 + 1) * • (1 « (log2_min_coding_block_size_minus3 + 3)) In HEVC Encoder mode, the following restrictions apply. Note : for a frame height that is not an integer multiple of LCU : Kernel : on last LCU at frame's bottom edge, it must align to CU boundary. This applies to all size of LCU: 16x16, 32x32, 64x64. Kernel does not count/include CUs outside of the picture in PakObj/CU_record respectively. Driver : sets up a LCU aligned (both in X/Y direction) surface.
	15	**Reserved**
	14:10	**Reserved** Format: MBZ
	9:0	**FrameWidthInMinCbMinus1** Format: U10 Specifies the width of each decoded picture in units of minimum coding block size. Value: [0-4122]; Name: **Programming Notes** The decoded picture width in units of luma samples equals • (FrameWidthInMinCbMinus1 + 1) * • (1 « (log2_min_coding_block_size_minus3 + 3))
2	15	**Reserved** Format: MBZ
	31:12	**Reserved** Format: MBZ
	11:10	**MaxPCMSize** Format: U2 Specifies the largest allowed PCM coding block size. 3 – Reserved 2 – 32x32 1 – 16x16 0 – 8x8

HCP_PIC_STATE

9:8 MinPCMSize

Format:	U2

Specifies the smallest allowed PCM coding block size.

Value	Name
3	Reserved
2	32x32
1	16x16
0	8x8

7:6 MaxTUSize

Format:	U2

Specifies the largest allowed transform block size.

Value	Name
3	32x32
2	16x16
1	8x8
0	4x4

5:4 MinTUSize

Format:	U2

Specifies the smallest allowed transform block size.

Value	Name
3	32x32
2	16x16
1	8x8
0	4x4

3:2 CtbSize (LCUSize)

Format:	U2

Specifies the coding tree block size.

Value	Name	Programming Notes
3	64x64	
2	32x32	
1	16x16	
0	illegal/reserved	

Programming Notes
LCU is restricted based on the picture size.

1:0 MinCUSize

HCP_PIC_STATE

		Format: U2 Specifies the smallest coding block size. Value / Name: 3 – 64x64 2 – 32x32 1 – 16x16 0 – 8x8
3	31:3	**Reserved** Format: MBZ
	2	**InsertTestFlag** Format: U1 Value / Name: 0h – **[Default]** 1h –

		Programming Notes
		CABAC 0 Word Insertion Test Enable (Encoder Only)This bit will modify CABAC K equation so that a positive K value can be generated easily. This is done for validation purpose only. In normal usage this bit should be set to 0.
		Regular equation for generating 'K' value when CABAC 0 Word Insertion Test Enable is set to 0.
		`K = {[((96 * pic_bin_count()) - (RawMinCUBits * PicSizeInMinCUs *3) + 1023) / 1024] - bytes_in_picture} / 3`
		Modified equation when CABAC 0 Word Insertion Test Enable bit set to 1.
		`K = {[((1536 * pic_bin_count()) - (RawMinCUBits * PicSizeInMinCUs *3) + 1023) / 1024] - bytes_in_picture} / 3`
		Encoder only feature.
		This bit should be set to 0 in regular PAK mode.

HCP_PIC_STATE

	1	**CurPicIsI** Format: U1 Specifies that the current picture is comprised solely of I slices and that there are no P or B slices in the picture. Value: 0 — Name: Current picture has at least one P or B slice **Programming Notes** This bit should be set to "0". Note: The value of "1" setting ("Current picture is comprised solely of I slices") is REMOVED. This bit is used for hardware optimization only. There is not enough information to set this bit to "1" correctly.
	0	**ColPicIsI** Format: U1 Specifies that the collocated picture is comprised solely of I slices and that there are no P or B slices in the picture. Value: 0 — Name: Collocated picture has at least one P or B slice **Programming Notes** This bit should be set to "0". Note: The value of "1" setting ("Collocated picture is comprised solely of I slices") is REMOVED. This bit is used for hardware optimization only. There is not enough information to set this bit to "1" correctly.
4	31:28	**Reserved** Format: MBZ
	26	**strong_intra_smoothing_enable_flag** Format: U1
	25	**transquant_bypass_enable_flag** Format: Enable Value 0 — Name: Disable — Description: cu_transquant_bypass is not supported Value 1 — Name: Enable — Description: cu_transquant_bypass is supported
	24	**Reserved** Format: MBZ

HCP_PIC_STATE

23 **amp_enabled_flag**

Format:	Enable

Value	Name	Description
0	Disable	Asymmetric motion partitions cannot be used in coding tree blocks.
1	Enable	Support asymmetric motion partitions, i.e. PartMode equal to PART_2NxnU, PART_2NxnD, PART_nLx2N, or PART_nRx2N.

22 **transform_skip_enabled_flag**

Format:	Enable

Value	Name	Description
0	Disable	transform_skip_flag is not supported in the residual coding
1	Enable	transform_skip_flag is supported

21 **BottomField**

Format:	U1

Value	Name	Description
0	Bottom Field	
1	Top Field	

Programming Notes
Must be zero for encoder only

20 **FieldPic**

Format:	U1

Value	Name	Description
0	Video Frame	
1	Video Field	

Programming Notes
Must be zero for encoder only.

19 **weighted_pred_flag**

Format:	U1

18 **weighted_bipred_flag**

Format:	U1

17 **tiles_enabled_flag**

HCP_PIC_STATE

	Format:	U1

Programming Notes
Tiling is not supported and this bit should be set to 0.

16 entropy_coding_sync_enabled_flag

Format:	U1

Not used in encoder mode

15 Reserved

15 loop_filter_across_tiles_enabled_flag

Format:	U1

14 Reserved

Format:	MBZ

13 sign_data_hiding_flag

Format:	Enable

Value	Name	Description
0	Disable	Specifies that sign bit hiding is disabled.
1	Enable	Specifies that sign bit hiding is enabled.

Programming Notes
Currently not supported in encoder, so must be set to 0 for encoding session.

12:10 log2_parallel_merge_level_minus2

Format:	U3

Value	Name	Programming Notes
[0,4]	Valid Range	The value of log2_parallel_merge_level_minus2 shall be in the range of 0 to Log2CtbSizeYCtbLog2SizeY - 2, inclusive.

Programming Notes
For encoder, always set to 0 (Intel restriction).

9 constrained_intra_pred_flag

Format:	U1

8 pcm_loop_filter_disable_flag

Format:	U1

7:6 diff_cu_qp_delta_depth (or named as max_dqp_depth)

HCP_PIC_STATE

		Format: U2
		Programming Notes cu_qp_delta_enabled_flag/max_dqp_depth: 1/0: has cu qp delta. (cu depth <= max_dqp_depth) will have cu qp delta coded. Only allow QP change across Slices, no change across LCU or CU.
	5	**cu_qp_delta_enabled_flag** Format: U1

Value	Name	Description
0	Disable	Does not allow QP change at CU or LCU level, the same QP is used for the entire slice. Max_DQP_Level = 0 (i.e. diff_cu_qp_delta_depath = 0).
1	Enable	Allow QP change at CU level. MAX_DQP_level can be >0.

	4	**pcm_enabled_flag** Format: U1
	3	**sample_adaptive_offset_enabled_flag** Format: U1
	2:0	**Reserved** Format: MBZ
5	31:30	**Reserved** Format: MBZ
	29:27	**bit_depth_luma_minus8** Format: U3 This specifies the number of bit allow for Luma pixels. In 8 bit mode, this must be set to 0. Encoder: Suports bit depths 8, 10 and 12 only. Encoder: Does not support 10 or 12 bit Source Pixels and 8bit PAK i.e. the source pixel depth should be less than or equal to PAK bit depth.

Value	Name	Description
0	luma_8bit	
1	luma_9bit	Only HEVC decoder supports 9 bits luma. HEVC encoder does not supports 9 bits luma.
2	luma_10bit	
3	luma_11bit	
4	luma_12bit	

	26:24	**bit_depth_chroma_minus8**

HCP_PIC_STATE

		Format: U3 This specifies the number of bit allow for Chroma pixels. In 8 bit mode, this must be set to 0. Encoder: Supports bit depths 8, 10 and 12 only. And also it must be same as Luma. Encoder: Does not support 10 or 12 bit Source Pixels and 8bit PAK. i.e. The source pixel depth should be less than or equal to the PAK bit depth. **Value / Name / Description** 0 chroma_8bit 1 chroma_9bit — Only HEVC decoder supports 9 bits chroma. HEVC encoder does not supports 9 bits chroma. 2 chroma_10bit 3 chroma_11bit 4 chroma_12bit
	23:20	**pcm_sample_bit_depth_luma_minus1** Format: U4
	19:16	**pcm_sample_bit_depth_chroma_minus1** Format: U4
	15:13	**max_transform_hierarchy_depth_inter(or named as tu_max_depth_inter)** Format: U3 Maximum TU split depths for inter blocks **Programming Notes** For encoder, always set to 2 to allow max 2 levels of split. For more splitting rely on CU split to match the content (Intel restriction)
	12:10	**max_transform_hierarchy_depth_intra (or named as tu_max_depth_intra)** Format: U3 Maximum TU split depth for intra blocks. **Programming Notes** For encoder, always set to 2 to allow max 2 levels of split. For more splitting, rely on CU split to match the content (Intel restriction).
	9:5	**pic_cr_qp_offset** Format: S4 Valid range -12 to +12
	4:0	**pic_cb_qp_offset** Format: S4 Valid range -12 to +12
6	31:30	**Reserved** Format: MBZ
	29	**Load Slice Pointer Flag**

HCP_PIC_STATE

Format:	Enable

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
0	Disable	Load BitStream Pointer only once for the first slice of a frame.
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.

Programming Notes
Must be zero for encoder

28:27 **Reserved**

Format:	MBZ

26 **FrameSzUnderStatusEn - FrameBitRateMinReportMask**

Format:	Enable

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

Value	Name	Description
0	Disable	Do not update bit 2 (Frame Bit Count Violate -- under run) of HCP_IMAGE_STATUS control register. NOTE: This bit MUST BE set to zero for the last BRC pass if SAO first pass also enabled. This will ensure that HW picks up right accumulated deltaQP for SAO sencond pass.
1	Enable	Set bit 2 (Frame Bit Count Violate -- under run) of HCP_IMAGE_STATUS control register if the total frame level bit counter is less than or equal to Frame Bit Rate Minimum limit. HW does not use this bit to set the bit in HCP_IMAGE_STATUS_CONTROL register. It's used pass the bit in HCP_IMAGE_STATUS_MASK register

Programming Notes
Encoder Only

25 **FrameSzOverStatusEn - FrameBitRateMaxReportMask**

Format:	Enable

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

HCP_PIC_STATE

Value	Name	Description
0	Disable	Do not update bit 1 of HCP_IMAGE_STATUS control register. NOTE: This bit MUST BE set to zero for the last BRC pass if SAO first pass also enabled. This will ensure that HW picks up right accumulated deltaQP for SAO sencond pass.
1	Enable	Set bit 1 of HCP_IMAGE_STATUS control register if the total frame level bit counter is greater than or equal to Frame Bit Rate Maximum limit. HW does not use this bit to set the bit in HCP_IMAGE_STATUS_CONTROL register. It's used pass the bit inHCP_IMAGE_STATUS_MASK register

Programming Notes
Encoder Only

24 **LCUMaxBitStatusEn - LCUMaxSizeReportMask**

Format:	Enable

This is a mask bit controlling if the condition of any LCU in the frame exceeds LCUMaxSize.

Value	Name	Description
0	Disable	Do not update bit 0 of HCP_IMAGE_STATUS control register. NOTE: This bit MUST BE set to zero for the last BRC pass if SAO first pass also enabled. This will ensure that HW picks up right accumulated deltaQP for SAO second pass.
1	Enable	Set bit 0 of HCP_IMAGE_STATUS control register if the total bit counter for the current LCU is greater than the LCU Conformance Max size limit. HW does not use this bit to set the bit in HCP_IMAGE_STATUS_CONTROL register.

Programming Notes
Encoder Only

23:17 **Reserved**

Format:	MBZ

16 **NonFirstPassFlag**

Format:	Enable

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

Value	Name	Description
0	Disable	If it is initial-Pass, this bit is set to 0.
1	Enable	For subsequent passes, this bit is set to 1.

HCP_PIC_STATE

		Programming Notes Encoder Only
	15:0	**LCU Max BitSize Allowed** Format: U16 This field specifies the Max LCU Bit Size allowed in according to the spec conformance. However, driver can program a different value from the spec for other encoding purpose. If LCU Limit exceeds, LCUMaxBitStatus in MMIO would be set to 1 and also in CU Statistics Streamout, LCU Limit exceed field would be set to 1 at the last CU of a LCU **Programming Notes** Encoder Only
7 **Programming Notes:** Encoder Only	31	**FrameBitrateMaxUnit** Format: U1 This field is the Frame Bitrate Maximum Limit Units. **Value** / **Name** / **Description** 0 / Byte / 32byte unit 1 / Kilo Byte / 4kbyte unit **Programming Notes** Encoder Only
	30:14	**Reserved** Format: MBZ Reserved for future expansion of the Max Rate.
	13:0	**FrameBitRateMax** Format: U14 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. **Value** / **Name** / **Description** 0-512KB / / The programmable range is 0-512KB when FrameBitrateMaxUnit is 0. 0-64MB / / The programmable range is 0-64Mbyte when FrameBitrateMaxUnit is 1.
8	31	**FrameBitrateMinUnit** Format: U1 This field is the Frame Bitrate Minimum Limit Units.

HCP_PIC_STATE

Value	Name	Description
0	Byte	32byte unit
1	Kilo Byte	4kbyte unit

Programming Notes
Encoder Only

30:14 **Reserved**

Format:	MBZ

Reserved for future expansion of the Min Rate.

13:0 **FrameBitRateMin**

Format:	U14

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 happen to be below this value.

Value	Name	Description
0-512KB		The programmable range is 0-512KB when FrameBitrateMinUnit is 0.
0-64MB		The programmable range is 0-64Mbyte when FrameBitrateMinUnit is 1.

Programming Notes
Encoder Only

9 31 **Reserved**

Format:	MBZ

30:16 **FrameBitRateMaxDelta**

Exists If:	//Always
Format:	U15

This field is used to select the slice delta QP when FrameBitRateMax Is exceeded. It shares the same FrameBitrateMaxUnit.

Value	Name	Description
0	**[Default]**	
0-1024KB		The Programmable range is 0-1024KB when FrameBitRateMaxUnit is 0.
0-128MB		The Programmable range is 0-128MB when FrameBitRateMaxUnit is 1.

Programming Notes

HCP_PIC_STATE

		Encoder Only
	15	**Reserved** Format: MBZ
	14:0	**FrameBitRateMinDelta** Format: U15 This field is used to select the slice delta QP when FrameBitRateMin Is exceeded. It shares the same FrameBitrateMinUnit. **Value** / **Name** / **Description** 0 / **[Default]** / 0-1024KB / / The Programmable range is 0-1024KB when FrameBitRateMinUnit is 0. 0-128MB / / The Programmable range is 0-128MB when FrameBitRateMinUnit is 1. **Programming Notes** HW requires the following condition: FrameBitRateMinDelta <= 2*FrameBitRateMinMust be true, otherwise it may cause unpredicted behavior. Encoder Only
10..11	63:0	**FrameDeltaQpMax** Format: FrameDeltaQp Range: [0:MAX_QP_DELTA] Frame level delta QP which should be used in case FrameSize - FrameBitRateMax in the range of ((DeltaQpMaxRange[n] * FrameBitRateMaxDelta»5)), DeltaQpMaxRange[n+1] * FrameBitRateMaxDelta»5)). **Programming Notes** If n == 7, DeltaQpMaxRange is infinity. Format: 8 bit sign-magnitude, Range: [0:63] Encoder Only

HCP_PIC_STATE

12..13	63:0	**FrameDeltaQpMin** Format: FrameDeltaQp Range: [0:MIN_QP_DELTA] Frame level delta QP which should be used in case FrameSize - FrameBitRateMin in the range of ((DeltaQpMinRange[n] * FrameBitRateMinDelta»5)), DeltaQpMinRange[n+1] * FrameBitRateMinDelta»5)). **Programming Notes** If n == 7, DeltaQpMinRange is infinity. Format: 8 bit sign-magnitude Range: [-63:0] Encoder Only
14..15	63:0	**FrameDeltaQpMaxRange** Format: FrameDeltaQpRange Range: [0:U8_MAX] Condition: FrameDeltaQpMaxRange[n] >= FrameDeltaQpMaxRange[n-1] This field is to calculate ranges for Frame level delta QP, specifically Frame level delta QP[n] and Frame level delta QP[n+1]. **Programming Notes** If n == 0, FrameDeltaQpMaxRange is zero. Encoder Only
16..17	63:0	**FrameDeltaQpMinRange** Format: FrameDeltaQpRange Range: [0:U8_MAX] Condition: FrameDeltaQpMinRange[n] >= FrameDeltaQpMinRange[n-1] This field is to calculate ranges for Frame level delta QP, specifically Frame level delta QP[n] and Frame level delta QP[n+1]. **Programming Notes** If n == 0, FrameDeltaQpMinRange is zero. Encoder Only

HCP_PIC_STATE

18	31:30	**MinFrameSizeUnits** Format: U2 This field is the Minimum Frame Size Units Value / Name / Description: 0 / 4Kb / Minimum Frame Size is in 4Kbytes. 1 / 16Kb / Minimum Frame Size is in 16Kbytes. 2 / Reserved / 3 / Reserved / **Programming Notes** Encoder Only
	29:16	**Reserved** Format: MBZ
	15:0	**MinFrameSize** Default Value: 0 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. It is needed for CBR. 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. This field is reserved in Decode mode. **Programming Notes** Programmable range is 0..(2^16-1) * 2^12 when MinFrameSizeUnits is 0. (4KB unit) Programmable range is 0..(2^16-1) * 2^14 when MinFrameSizeUnits is 1. (16KB unit) Encoder Only

HCP_PIPE_MODE_SELECT

HCP_PIPE_MODE_SELECT

Source: VideoCS

Length Bias: 2

The HCP is selected with the **Media Instruction Opcode "7h"** for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.

The workload for the HCP is based upon a single frame decode. There are no states saved between frame decodes in the HCP. Once the bit stream DMA is configured with the HCP_BSD_OBJECT command, and the bit stream is presented to the HCP, the frame decode will begin.
The HCP_PIPE_MODE_SELECT command is responsible for general pipeline level configuration that would normally be set once for a single stream encode or decode and would not be modified on a frame workload basis.
This is a picture level state command and is shared by both encoding and decoding processes.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: 0h HCP_PIPE_MODE_SELECT Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n (Excludes Dwords 0, 1). Value: 4h — Name: Value_4
1	31:24	**Reserved**
	23	**Reserved** Format: MBZ

HCP_PIPE_MODE_SELECT

22:20	**Reserved**	
	Format:	MBZ
19:18	**Reserved**	
	Format:	MBZ
17	**RESERVED**	
	Format:	MBZ
16:13	**Reserved**	
	Format:	MBZ
12	**Reserved**	
	Format:	MBZ
11	**Reserved**	
	Format:	MBZ
10	**Reserved**	
9	**Reserved**	
8	**Reserved**	
7:5	**Codec Standard Select**	
	Value	**Name**
	0	HEVC
4	**Reserved** This bit is reserved since it is used by HUC_PIPE_MODE_SELECT. Making sure there is no overlap between the two commands.	
3	**Pic Status/Error Report Enable**	
	Format:	Enable

Value	Name	Description
0	Disable	Disable status/error reporting
1	Enable	Status/Error reporting is written out once per picture. The Pic Status/Error Report ID in DWord3 along with the status/error status bits are packed into one cache line and written to the Status/Error Buffer address in the HCP_PIPE_BUF_ADDR_STATE command. Must be zero for encoder mode.

2	**Reserved**	
1	**Deblocker Streamout Enable**	
	Format:	Enable

Deblocker Streamout Enable not currently supported for Encode or Decode

Value	Name	Description
0	Disable	Disable deblocker-only parameter streamout
1	Enable	Enable deblocker-only parameter streamout

HCP_PIPE_MODE_SELECT

	0	**Codec Select** Format: U1 Value / Name 0 – Decode 1 – Encode
2	31:0	**Media Soft-Reset Counter (per 1000 clocks)** Format: U32 In decoder modes, this counter value specifies the number of clocks (per 1000) of GAC inactivity before a media soft-reset is applied to the HCP and HuC. If counter value is set to 0, the media soft-reset feature is disabled and no reset will occur. In encoder modes, this counter must be set to 0 to disable media soft reset. This feature is not supported for the encoder. Value / Name 0 – Disable
3	31:0	**Pic Status/Error Report ID** Format: U32 The Pic Status/Error Report ID is a unique 32-bit unsigned integer assigned to each picture status/error output. Must be zero for encoder mode. Value / Name / Description 0 – 32-bit unsigned – Unique ID Number 1 – Reserved **Programming Notes** Software must program different Status/Error Buffer addresses between pictures; otherwise the hardware might overwrite previously written data.

HCP_SLICE_STATE

HCP_SLICE_STATE

Source: VideoCS

Length Bias: 2

The HCP is selected with the **Media Instruction Opcode "7h"** for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.

This is a slice level command used in both encoding and decoding processes. For decoder, it is issued with the HCP_BSD_OBJECT command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: 14h HCP_SLICE_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 7h — Name:
1	31:25	**Reserved** Format: MBZ
	24:16	**SliceStartCtbY or (slice_start_lcu_y encoder)** Format: U9 Specifies the starting row address of the first coding tree block in the current slice.
	15:9	**Reserved** Format: MBZ

HCP_SLICE_STATE

	8:0	**SliceStartCtbX or (slice_start_lcu_x encoder)** Format: U9 Specifies the starting column address of the first coding tree block in the current slice.
2	31:25	**Reserved** Format: MBZ
	24:16	**NextSliceStartCtbY or (next_slice_start_lcu_y encoder)** Format: U9 Specifies the starting row address of the first coding tree block in the next slice. Must be set to zero when the current slice is the last slice of a picture. For the single slice per frame case, the only slice is also the last slice, so this parameter should be set to a number larger than the frame height (at least +1).
	15	**Reserved**
	14:9	**Reserved** Format: MBZ
	8:0	**NextSliceStartCtbX or (next_slice_start_lcu_x encoder)** Format: U9 Specifies the starting column address of the first coding tree block in the next slice. Must be set to zero when the current slice is the last slice of a picture. For the single slice per frame case, the only slice is also the last slice, so this parameter should be set to a number larger than the frame width (at least +1).
3	31:26	**Reserved** Format: MBZ
	25	**Reserved** Format: MBZ
	24	**Reserved** Format: MBZ
	23	**Reserved** Format: MBZ
	22	**Reserved**
	21:17	**slice_cr_qp_offset** Format: S4 For deblocking purpose, the pic and slice level cr qp offset must be provided separately. PAK needs to perform final_chroma_cr_qp_offset = pic_cr_qp_offset + slice_cr_qp_offset.

Value	Name
14h	-12
15h	-11

HCP_SLICE_STATE

16h	-10
17h	-9
18h	-8
19h	-7
1Ah	-6
1Bh	-5
1Ch	-4
1Dh	-3
1Eh	-2
1Fh	-1
0h	0
1h	1
2h	2
3h	3
4h	4
5h	5
6h	6
7h	7
8h	8
9h	9
0Ah	10
0Bh	11
0Ch	12

Programming Notes
The valid value is from -12 to 12 (or 14h to 0Ch).

16:12 **slice_cb_qp_offset**

Format:	S4

For deblocking purpose, the pic and slice level cb qp offset must be provided separately.
PAK needs to perform final_chroma_cb_qp_offset = pic_cb_qp_offset + slice_cb_qp_offset.

Value	Name
14h	-12
15h	-11
16h	-10

HCP_SLICE_STATE

17h	-9
18h	-8
19h	-7
1Ah	-6
1Bh	-5
1Ch	-4
1Dh	-3
1Eh	-2
1Fh	-1
0h	0
1h	1
2h	2
3h	3
4h	4
5h	5
6h	6
7h	7
8h	8
9h	9
0Ah	10
0Bh	11
0Ch	12

Programming Notes
The valid value is from -12 to 12 (or 14h to 0Ch).

11:6 **SliceQp**

Format:	U6

Specifies the initial absolute value of QPy quantization parameter for the slice as defined in the Slice Header Semantics section of the HEVC standard.
This signifies only the magnitude of SliceQp. In 8 bit, SliceQp only goes from 0 to 51. But in 10 bit, it needs to go from -12 to 51. There is a sign bit specifies at bit [3] below.

5 **slice_temporal_mvp_enable_flag**

Format:	U1

Programming Notes
Must be same for all the slices within a frame in encoder mode (follow spec)

HCP_SLICE_STATE

	4	**dependent_slice_flag** Format: U1 Decoder only.
	3	**SliceQp Sign Flag** Format: U1 This specifes the sign bit of SliceQp. This is added for HEVC 10 bit. For 8 bit, SliceQp goes from 0 to 51 so this bit should be zero. In 10 bit, SliceQp goes from -12 to 51 and this bit can be set for negative value.
	2	**LastSliceofPic** Format: U1 This indicates the current slice is the very last slice of the current picture

Value	Name
0	Not the last slice of the picture
1	Last slice of the picture

	1:0	**slice_type** Format: U2

Value	Name
0	B-slice
1	P-slice
2	I-slice
3	Illegal/Reserved

4	31:29	**Reserved** Format: MBZ
	28:26	**CollocatedRefIDX** Format: U3 Collocated Motion Vector Temporal Buffer Index.
	25:23	**MaxMergeIDX** Format: U3 MaxNumMergeCand = 5 - five_minus_max_num_merge_cand -1.

Value	Name
0	0
1	1
2	2
3	3
4	4

HCP_SLICE_STATE

		Programming Notes The valid value is from 0 to 4 (MaxNumMergeCand = 5 - five_minus_max_num_merge_cand -1)
	22	**cabac_init_flag** Format: U1
	21:19	**luma_log2_weight_denom** Format: U3
	18:16	**ChromaLog2WeightDenom** Format: U3
	15	**collocated_from_l0_flag** Format: U1
	14	**isLowDelay** Format: U1 If the POCs of all pictures in both lists are less than the current POC, then set to one, else set to zero.
	13	**mvd_l1_zero_flag** Format: U1 Decoder only.
	12	**slice_sao_luma_flag** Format: U1
	11	**slice_sao_chroma_flag** Format: U1
	10	**slice_loop_filter_across_slices_enabled_flag** Format: U1
	9	**Reserved** Format: MBZ
	8:5	**slice_beta_offset_div2 or (final Beta_Offset_div2 Encoder)** Format: S3 Deblocking filter beta offset. Specified in 2's comp. **Value** [1101b,0011b] — **Name** [-3,3] **Programming Notes** Valid only in encoder mode
	4:1	**slice_tc_offset_div2 or (final tc_offset_div2 Encoder)** Format: S3

HCP_SLICE_STATE

Deblocking filter tc offset. Specified in 2's comp.

Value	Name
[1101b,0011b]	[-3,3]

Programming Notes
Valid only in encoder mode

slice_header_disable_deblocking_filter_flag

Format:	U1

Reserved

Format:	MBZ

SliceHeaderLength

Format:	U16

Decoder only.

Specifies the length in bytes of the slice header including the start code. The starting byte of the slice header in the bit stream buffer is indicated by the Indirect Data Start Address in the HCP_BSD_OBJECT command. The ending byte of the slice header in the same bit stream buffer is indicated by the last byte prior to the slice data (CABAC).

Reserved

Format:	MBZ

RoundInter

Format:	U4

Value	Name
0h	+1/32
1h	+2/32
2h	+3/32
3h	+4/32
4h	+5/32 **[Default]**
5h	+6/32
6h	+7/32
7h	+8/32
8h	+9/32
9h	+10/32
Ah	+11/32
Bh	+12/32
Ch	+13/32

HCP_SLICE_STATE

		Dh: +14/32 Eh: +15/32 Fh: +16/32 **Programming Notes** Encoder only feature
	25:24	**Reserved** Format: MBZ
	23:20	**RoundIntra** Format: U4 **Value** / **Name** 0h: +1/32 1h: +2/32 2h: +3/32 3h: +4/32 4h: +5/32 **[Default]** 5h: +6/32 6h: +7/32 7h: +8/32 8h: +9/32 9h: +10/32 Ah: +11/32 Bh: +12/32 Ch: +13/32 Dh: +14/32 Eh: +15/32 Fh: +16/32 **Programming Notes** Encoder only feature
	19:0	**Reserved** Format: MBZ
7	31:11	**Reserved** Format: MBZ

HCP_SLICE_STATE

10 Header Insertion Enable

Format:	U1

Must be followed by the PAK Insertion Object Command to perform the actual insertion.

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

Programming Notes
Must be always enabled. Encoder Only feature

9 SliceData Enable

Format:	U1

Must always be enabled.
Encoder only feature.

Value	Name	Description
0		No operation; no insertion.
1		Slice Data insertion by PAK Object Commands into the output bitstream buffer.

8 Tail Insertion Enable

Format:	U1

Must be followed by the PAK Insertion Object Command to perform the actual insertion.

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.

Programming Notes
Tail Insertion is allowed only at the end of last slice or last tile of a frame but not in the middle of frame. Also, no multiple tail insertions are allowed. Applies to all projects starting from SNB+ Encoder only feature

7:3 Reserved

Format:	MBZ

HCP_SLICE_STATE

	2	**EmulationByteSliceInsertEnable** Format: U1 To have PAK outputting SODB or EBSP to the output bitstream buffer. Value / Name: 0 outputting RBSP 1 outputting EBSP **Programming Notes** Encoder Only feature
	1	**CabacZeroWordInsertionEnable** Format: U1 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). **Programming Notes** Encoder Only feature
	0	**Reserved** Format: MBZ
8	31:29	**Reserved** Format: MBZ
	28:6	**Indirect PAK-BSE Data Start Offset (Write)** Format: U23 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 HCP PAK-BSE Object Base Address. It is a cacheline-aligned address for the HEVC bitstream data. For Write, there is no need to have a data length field. It is assumed the global memory upper 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

HCP_SLICE_STATE		
		Programming Notes
		Must be zero.
		Encoder Only feature
	5:0	**Reserved**
		Format: MBZ

HCP_SURFACE_STATE

HCP_SURFACE_STATE

Source: VideoCS

Length Bias: 2

The HCP is selected with the **Media Instruction Opcode "7h"** for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.

The HCP_SURFACE_STATE command is responsible for defining the frame buffer pitch and the offset of the chroma component.
This is a picture level state command and is shared by both encoding and decoding processes.
Note : Only NV12 and Tile Y are being supported for HEVC. Hence full pitch and interleaved UV is always in use. U and V Xoffset must be set to 0; U and V Yoffset must be 16-pixel aligned. This Surface State is not the same as that of the 3D engine and of the MFX pipeline.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: 1h HCP_SURFACE_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 1h — Name:
1	31:28	**Surface Id** Format: U4 Value / Name / Description

HCP_SURFACE_STATE

0h	HEVC: For current decoded Picture	8-bit uncompressed data
1h	Source Input Picture (encoder)	8-bit uncompressed data
2h	Reserved	
3h	Reserved	
4h	Reserved	

27:17 Reserved

Format:	MBZ

16:0 Surface Pitch Minus1

Format:	U17-1

This field specifies the surface pitch in (#Bytes - 1).

Programming Notes

For TileYF and TileYS surfaces, the range is dependent on the Cu parameter (refer to Memory Data Formats section for the definition of the Cu parameter depending on the case).
The range in bytes is [2^{Cu}-1, 131071] -> [(2^{Cu})B, 128KB] = [1 tile, 128KB/($2^{Cu)\ tiles}$]
The field specifies the surface pitch in (#Bytes - 1)

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, 131071] to [128B,128KB] = [1 tile, 1024 tiles]

2 31:28 Surface Format

Format:	U4

Specifies the format of the surface.

Value	Name
0h-2h	Reserved
4h	PLANAR_420_8
5h-Ch	Reserved
Dh	P010
Fh	Reserved

27:15 Reserved

Format:	MBZ

14:0 Y Offset for U(Cb) in pixel

Format:	U15_Pixel_Row_Offset

This field specifies the vertical 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

The following restrictions are applicable when Memory compression is enabled.

HCP_SURFACE_STATE		
		• For PLANAR_420 surface formats, the alignment of this field follows the tile mode described in bits 14:13 of the **Memory Address Attributes** table. • TileY (legacy 4k) - 8 pixel aligned • TileYF (New 4k) - 64 pixel aligned • TileYS (64k) - 256 pixel aligned
		When Memory compression is not enabled, This field should be multiple of 8 pixels.

HUC_PIPE_MODE_SELECT

HUC_PIPE_MODE_SELECT

Source: VideoCS

Length Bias: 2

The HUC is selected with the Media Instruction Opcode "Bh" for all HUC Commands. Each HUC command has assigned a media instruction command as defined in DWord 0, BitField 22:16.

The HUC_PIPE_MODE_SELECT command is responsible for general pipeline level configuration that would normally be set once for a single stream decode and would not be modified on a frame workload basis.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: Bh Codec/Engine Name Format: OpCode Codec/Engine Name = HUP = Bh
	22:16	**Media Instruction Command** Default Value: 0h HUC_PIPE_MODE_SELECT Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n (Excludes Dwords 0, 1). Value: 1h — Name:
1	31:24	**Reserved**
	23:11	**Reserved** Format: MBZ
	10	**HUC Stream Object Enable** This indicates that HUC Stream Object command is going to be programmed. This bit should be set to "1" if HUC_STREAM_OBJECT command is going to be programmed. If this bit is not set, HUC_STREAM_OBJECT should not be programmed in the current operation. This bit is only used by hardware to prepare for bitstream processing.

HUC_PIPE_MODE_SELECT

	9:5	**Reserved** Format: MBZ
	4	**Indirect Stream Out Enable** Format: Enable Enables the bitstream to be written out to memory. The memory buffer is addressed through the HuC Indirect Stream Out ObjectBase Address. **Value** / **Name** 0h: Disable Indirect Stream Out 1h: Enable Indirect Stream Out
	3:0	**Reserved** Format: MBZ
2	31:0	**Media Soft-Reset Counter (per 1000 clocks)** Format: U32 In decoder modes, this counter value specifies the number of clocks (per 1000) of GAC inactivity before a media soft-reset is applied to the HCP and HuC. If counter value is set to 0, the media soft-reset feature is disabled and no reset will occur. In encoder modes, this counter must be set to 0 to disable media soft reset. This feature is not supported for the encoder. **Value** / **Name** 0: Disable
3	31:0	**Reserved** This field is intentionally left blank to match the format of HCP_PIPE_MODE_SELECT.
4	31:0	**Reserved**
5	31:0	**Reserved**

HUC_STREAM_OBJECT

HUC_STREAM_OBJECT

Source: VideoCS

Length Bias: 2

The HUC is selected with the Media Instruction Opcode "Bh" for all HUC Commands. Each HUC command has assigned a media instruction command as defined in DWord 0, BitField 22:16.

The HUC_STREAM_OBJECT command is used to define the bit stream address offset to the Stream Indirect Object base Address and the length of the bit stream. The bitstream buffer the HUC operates upon is specified through indirect addressing.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: Bh Codec/Engine Name Format: OpCode Codec/Engine Name = HUC = Bh
	22:16	**Media Instruction Command** Default Value: 20h HUC_STREAM_OBJECT Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 3h — Name:

HUC_STREAM_OBJECT

1 — 31:0 — Indirect Stream In Data Length

Format:	U32

Specifies the length in bytes of the bit stream input data.

Value	Name	Description
[0,268435455]	Data_Length_with_28_bits_only	Valid range is only from 0 to 268435455, which is corresponding to lower 28 bits. This restriction is for old project which only use 28 bits data length.
[268435456,2147483547]	Data_Length_beyond_28_bits	This is an added valid range when the bit length is extended from 28 bits to 32 bits. This is added to support 16k x 16k picture size bitstream.

2 — 31 — Reserved

2 — 30:29 — Reserved

Format:	MBZ

2 — 28:0 — Indirect Stream In Start Address

Format:	U29

Specifies the byte-aligned graphics memory starting address of the input bit stream relative to the **HUC Indirect Stream In ObjectBase Address [31:12]**.

3 — 31:29 — Reserved

Format:	MBZ

3 — 28:0 — Indirect Stream Out Start Address

Format:	U29

Specifies the byte-aligned graphics memory starting address of the output bit stream relative to the **HUC Indirect Stream Out ObjectBase Address [31:12]**.

4 — 31:30 — Reserved

Format:	MBZ

4 — 29 — HuC Bitstream Enable

Format:	Enable

Enables the bitstream to be sent to the HuC

Value	Name
0h	Disable
1h	Enable

HUC_STREAM_OBJECT

28:27 **DRMLengthMode**

Format:	U2

Value	Name	Description
00b	Start Code Mode	Stops on a start code
01b	Length Mode	Stops after a number of bytes are reached in the length counter
10b	Reserved	
11b	Reserved	

26 **Reserved**

25 **Emulation Prevention Byte Removal**

Format:	Enable

Value	Name	Description
0	Disable	Bypass Emulation Prevention Byte Removal.
1	Enable	Emulation prevention bytes will be removed after the start code search engine.

24 **Start Code Search Engine**

Format:	Enable

Value	Name	Description
0	Disable	Bypass Start Code Search Engine
1	Enable	Enables the start code search engine to stop on every third byte start code defined by **Start Code Byte [2:0]** defined in this DWord.

23:16 **Start Code Byte [2]**

Format:	U8

Third byte of the start code

15:8 **Start Code Byte [1]**

Format:	U8

Second byte of the start code

7:0 **Start Code Byte [0]**

Format:	U8

First byte of the start code

MEDIA_OBJECT_GRPID

MEDIA_OBJECT_GRPID

Source: RenderCS

Length Bias: 2

The MEDIA_OBJECT_GRPID command is a variation of MEDIA_OBJECT which includes a group id which is used to allocate and track Barriers and Shared Local Memory. The Interface Descriptor is used to specify how much SLM is needed and how many threads will be reporting to the Barrier. All MEDIA_OBJECT_GRPIDs with the same group id should have the same interface descriptor and be dispatched to the same Tslice – the dispatcher will ensure this if Force Destination = 0, but software must ensure this if Force Destination = 1. Software should also ensure that all the threads needed for the Barrier will fit into a Tslice, or the Barrier will never be satisfied. Either SLM or a barrier must be used with MEDIA_OBJECT_GRPID, if neither is needed then a MEDIA_OBJECT must be used instead.

MEDIA_OBJECT_GRPID supports the GPGPU version of payload delivery – either indirect or CURBE can be split between the threads in a group (per-thread payload), as well as a section which is sent to all threads (cross-thread payload). See the GPGPU payload section. For indirect, the same pointer must be sent with all the commands associated with the thread group for payload splitting to work properly. Inline data is not split, but the payload attached to each command is sent with that thread. Only one of inline, indirect, or CURBE is allowed, but at least one form of payload must be sent.

MEDIA_STATE_FLUSH with the watermark bit must be placed between groups created by MEDIA_OBJECT_GRPID. The Interface Descriptor associated with the watermark must match the Interface Descriptor used for the following group.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Media Command Pipeline** Default Value: 2h Media Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h MEDIA_OBJECT_GRPID Format: OpCode
	23:16	**Media Command Sub-Opcode** Default Value: 6h MEDIA_OBJECT_GRPID SubOp Format: OpCode
	15:0	**DWord Length** Default Value: 5h DWORD_COUNT_n Format: =n Total Length - 2 Excludes DWords 0,1 **Generic Mode:** DWord Length = N+5, where N is in the range of [0,504]. The maximum is 504 DW (equivalent to 63 8-DW registers).

MEDIA_OBJECT_GRPID

		When both inline and indirect data are fetched for this command, the total size in 8-DW registers must be less than 112 (with both inline data length N and indirect data length rounded up to 8-DW aligned individually). The minimal inline data length is 0.
1	31:8	**Reserved**
	7:6	**Reserved** Format: MBZ
	5:0	**Interface Descriptor Offset** Format: U6 This field specifies the offset from the interface descriptor base pointer to the interface descriptor which will be applied to this object. It is specified in units of interface descriptors. **Value** / **Name** [0,30]
2	31:25	**Reserved** Format: MBZ
	24	**Reserved**
	23	**End of Thread Group** This bit indicates that this dispatch is the last for the current thread group.
	22	**Force Destination** **Description** If set, bits 20:17 are used to determine the destination of this dispatch, if clear the destination will be chosen based on load. When using Force Destination=1 and Pooled EUs with BXT-A, the size of the maximum thread group using either barriers or SLM for subslice 1 is 18 rather than 36, since subslice 1 is split between 2 pools. If neither barriers nor SLM are used then the normal thread group size of 36 is allowed. BXT-C is not impacted and all 36 thread can be forced to either subslice.
	21	**Use Scoreboard** This field specifies whether the thread associated with this command uses hardware scoreboard. Only when this field is set, the scoreboard control fields in the VFE Dword are valid. If this field is cleared, the thread associated with this command bypasses hardware scoreboard. **Value** / **Name** 0 / Not using scoreboard 1 / Using scoreboard
	20:19	**Slice Destination Select** This bit along with the SubSlice destination select determines the slice that this thread must be sent to. Ignored if **Force Destination** = 0, or if product only has 1 slice. **Value** / **Name** / **Description** 00b / Slice 0

MEDIA_OBJECT_GRPID

01b	Slice 1	Cannot be used in products without a Slice 1.
10b	Slice 2	Cannot be used in products without a Slice 2.
11b	Reserved	

18:17 SubSlice Destination Select

This field selects the SubSlice that this thread must be sent to. Ignored if **Force Destination** = 0

Value	Name
11b	Subslice3
10b	SubSlice 2
01b	SubSlice 1
00b	SubSlice 0

16:0 Indirect Data Length

Format:	U17 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. This field must have the same alignment as the Indirect Object Data Start Address. It must be DQWord (32-byte) aligned. As the indirect data are sent directly to URB, range is limited to(URB Entry Allocation Size)*(Number of URB Entries) in the MEDIA_VFE_STATE command.

3 31:0 Indirect Data Start Address

Format:	GraphicsAddress[31:0]

Description
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 **Indirect Object Base Address**. Hardware ignores this field if indirect data is not present. Alignment of this address depends on the mode of operation.
It is the 64-byte aligned address of the indirect data.

Value	Name	Description
[0-512MB]		Bits 31:29 MBZ

4 31:25 Reserved

Format:	MBZ

24:16 Scoreboard Y

Format:	U9

This field provides the Y term of the scoreboard value of the current thread.

15:9 Reserved

Format:	MBZ

MEDIA_OBJECT_GRPID

	8:0	**Scoreboard X** Format: U9 This field provides the X term of the scoreboard value of the current thread.
5	31:20	**Reserved** Format: MBZ
	19:16	**Scoreboard Color** Format: U4 This field specifies which dependency color the current thread belongs to. It affects the dependency scoreboard control.
	15:8	**Reserved** Format: MBZ
	7:0	**Scoreboard Mask** Format: Boolean Each bit indicates the corresponding dependency scoreboard is dependent on. This field is AND'd with the corresponding Scoreboard Mask field in the MEDIA_VFE_STATE command. **Bit n (for n = 0…7):** Scoreboard n is dependent, where bit 0 maps to n = 0.
6	31:0	**GroupID** A unique identifying number which describes the threads which share a barrier and/or SLM. Reuse of numbers is allowed as long as the old group is not currently running.
7..n	31:0	**Inline Data** The format of this data is specified by software. Hardware does not interpret this data; it merely passes it to the kernel for processing. The total size for the inline data and indirect data must not exceed 112 registers.

MEDIA_POOL_STATE

MEDIA_POOL_STATE

Source: RenderCS

Length Bias: 2

The MEDIA_POOL_STATE command is used to enable pooled-mode and assign EUs to pools. This command applies to both GPGPU & media operations. If pooled operation is desired, this command must be performed prior to sending any work to the pool. Failure to send this command prior to work commands results in the default 'LegacyMode' operation which contains thread groups to single subslices.
The mechanism for assigning EUs to pools via a bitfield mapping as part of this command. Note that not all possible pool combinations available in the bit-mask may be supported on any given SKU and/or across architectural generations.
This command is ignored in 3D mode (as determined by pipeline choice in the most recent PIPELINE_SELECT command). This command sets context state which is saved/restored with any context switch. This command is normally only issued once with each context initialization.
PoolBitFields called for below map bit positions to individual EUs as follows:

- bit0 = row0, eu0
- bit1 = row0, eu1
- bit2 = row0, eu2
- bit3 = row0, eu3
- bit4 = row1, eu0
- bit5 = row2, eu1
- bit6 = row3, eu2
- bit7 = row4, eu3

Valid Pool Sizes and Configurations:
The table below identifies the valid pool configurations, depending on the subslice organization in the slice. There may be performance implications with the various settings depending on the size of the thread group as discussed in this feature's overview. The table ignores cases where one EU may be held back for manufacturability. For example, the 2x6 configuration may have a total of 12 or 11 valid EUs depending on SKUing, but only 12,0 is used in this command to cover both cases.
Note: For BXT 3x6 a pool must consist of a subslice and a row from the center subslice, while for BXT 2x6 a pool must consist of a complete subslice, so only 9,9 is valid for 3x6 and 12,0 is valid for 2x6.

PoolBitField				
BXT Config	**Slice 0**	**Slice 1**	**Slice 2**	**Slice 3**
2x6*	00000000h	0h	0h	0h
3x6*	00777000h	0h	0h	0h

** Includes cases where all EUs in the slice are available and cases where 1 EU in the slice is disabled.*

DWord	Bit	Description

MEDIA_POOL_STATE

0 — 31:29 — Command Type

Default Value:	3h GFXPIPE
Format:	OpCode

28:27 — Pipeline

Default Value:	2h Media
Format:	OpCode

26:24 — Media Command Opcode

Default Value:	0h
Format:	OpCode

23:16 — SubOpcode

Default Value:	5h MEDIA_POOL_STATE
Format:	OpCode

15:0 — DWord Length

Format:	=n

n = Total Length - 2

Value	Name	Description
04h	DWORD_COUNT_n **[Default]**	Excludes DWord (0,1)

1 — 31 — PoolEnable

Format:	U1

- When set to 1, this field enables the GPGPU Pool mechanism for all slices in the system.
- When set to 0, legacy mode is in effect and pools are not defined; instead thread groups will be dispatched in such a way that each is contained to a single subslice. This legacy mode is provided *primarily* for backward *driver* compatibility, and may *in some cases* be non-performant *due to thread loading or SLM bandwidth*.
- Mixing of LegacyMode and PooledMode across contexts is not allowed; i.e. the setting of the 'PooledMode' bit must be identical across all contexts until a subsequent HW reset.

Value	Name
0b	Disable **[Default]**
1b	Enable

30 — Reserved

Format:	MBZ

29:0 — Reserved

Format:	MBZ

2 — 31:0 — PoolBitField-Slice0

Format:	SubslicePool

MEDIA_POOL_STATE		
3	31:0	**PoolBitField-Slice1** Format: SubslicePool
4	31:0	**PoolBitField-Slice2** Format: SubslicePool
5	31:0	**PoolBitField-Slice3** Format: SubslicePool

MEDIA_STATE_FLUSH

MEDIA_STATE_FLUSH

Source: RenderCS

Length Bias: 2

This command updates the Message Gateway state. In particular, it updates the state for a selected Interface Descriptor.

This command can be considered same as a MI_Flush except that only media parser will get flushed instead of the entire 3D/media render pipeline. The command should be programmed prior to new Media state, curbe and/or interface descriptor commands when switching to a new context or programming new state for the same context. With this command, pipelined state change is allowed for the media pipe.

Be cautious when using this command when child_present flag in the media state is enabled. This is because that CURBE state as well as Interface Descriptor state are shared between root threads and child threads. Changing these states while child threads are generated on the fly may cause unexpected behavior. Combining with MI_ARB_ON/OFF command, it is possible to support interruptability with the following command sequence where interrupt may be allowed only when MI_ARB_ON_OFF is ON:

```
MEDIA_STATE_FLUSH
 VFE_STATE // VFE will hold CS if watermark isn't met
 MI_ARB_OFF // There must be at least one VFE command before this one
 MEDIA_OBJECT ... MI_ARB_ON
```

Workaround

Due to the known HW issue MEDIA_STATE_FLUSH command is not preemptable in GPGPU workloads. This might result in reduced frequency of preempting GPGPU walker in below programming sequence.

```
//State Commands
 GPGPU_WALKER
 MEDIA_STATE_FLUSH
 //State Commands
 GPGPU_WALKER
 MEDIA_STATE_FLUSH
 //State Commands
```

Below workaround must be applied to address the above issue.

MI_ARB_CHK command must be programmed following MEDIA_STATE_FLUSH command. PIPECONTROL ("Command Streamer Stall Enable" set and "Render Target Cache Flush Enable" set) must be programmed prior to MEDIA_STATE_FLUSH command with "Flush to GO" bit set. "Water Mark Required" field must not be set in "MEDIA_STATE_FLUSH" command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode

MEDIA_STATE_FLUSH

	26:24	**Media Command Opcode** Default Value: 0h MEDIA_STATE_FLUSH Format: OpCode
	23:16	**SubOpcode** Default Value: 4h MEDIA_STATE_FLUSH SubOp Format: OpCode
	15:0	**DWord Length** Format: =n Total Length - 2 **Value** / **Name** / **Description** 0h / DWORD_COUNT_n **[Default]** / Excludes DWord (0,1)
1	31:9	**Reserved** Format: MBZ
	8	**Reserved**
	7	**Flush to GO** Format: Enable This bit indicates that the write data out of this thread group should be flushed to the point where it is visible to following commands. **Workaround** Workaround: PIPECONTROL ("Command Streamer Stall Enable" set and "Render Target Cache Flush Enable" set) must be programmed prior to MEDIA_STATE_FLUSH command with "Flush to GO" bit set.
	6	**Watermark Required** This is a single bit specifying if the MEDIA_STATE_FLUSH should stall further commands until there is enough room in a half-slice for the following thread group. The characteristics of the thread group are specified in the Interface Descriptor Offset. If set, the MEDIA_STATE_FLUSH stalls CS until there are enough threads in a half-slice, and enough SLM available in the same half-slice, and a free barrier if one is required. An Interface Descriptors can be updated after a Watermarked MEDIA_STATE_FLUSH only if it has not been used in the current context. Reusing an interface desciptor requires that this bit is clear to ensure the ID cache is reloaded. If clear, the MEDIA_STATE_FLUSH stalls CS until the TDL has dispatched the last thread, allowing the CURBE and Interface Descriptors to be updated by following commands. **Programming Notes** The Interface Descriptor Offset used for the flush must be the same as that used for the GPGPU_OBJECTs. GPGPU_WALKER automatically checks the Watermark conditions before starting a thread, so this bit should not be set before GPGPU_WALKER. **Workaround**

MEDIA_STATE_FLUSH		
		Workaround: If pre-emption is used, the WatermarkRequired bit must not be set.
	5:0	**Interface Descriptor Offset** Format: U6 This field specifies the offset from the interface descriptor base pointer to the interface descriptor which describes what resources are required to meet the watermark.

MFD_VC1_LONG_PIC_STATE

MFD_VC1_LONG_PIC_STATE

Source: VideoCS

Length Bias: 2

MFX_VC1_LONG PIC_STATE command encapsulates the decoding parameters that are read or derived from bitstream syntax elements above (inclusive) picture header layer. These parameters are static for a picture and when slice structure is present, these parameters are not changed from slice to slice of the same picture. Hence, this command is only issued at the beginning of processing a new picture and prior to the VC1_*_OBJECT command. The values set for these state variables are retained internally across slices.Only the parameters needed by hardware (BSD unit) to decode bit sequence for the macroblocks in a picture layer or a slice layer are presented in this command. Other parameters such as the ones used for inverse transform or motion compensation are provided in MFX_VC1_PRED_PIPE_STATE command.This Long interface format is intel proprietary interface. Driver will need to perform addition operations to generate all the fields in 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_VC1_LONG_PIC_STATE Format: OpCode
	26:24	**Media Command Opcode** Default Value: 2h VC1_DEC Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h Format: OpCode
	20:16	**SubOpcode B** Default Value: 1h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 0004h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:24	**Reserved** Format: MBZ

MFD_VC1_LONG_PIC_STATE

	23:16	**PictureHeightInMBsMinus1 (Picture Height Minus 1 in Macroblocks)** Format: U8 This field indicates the height of the picture in unit of macroblocks. For example, for a 1920x1080 frame picture, PictureHeightInMBs equals 68 (1080 divided by 16, and rounded up, i.e. effectively specified as 1088 instead).This field is used in VLD and IT modes. **Value** / **Name** / **Description** 255 / Value_255 / **Programming Notes** Note: Even though the Advanced Profile allows frame dimensions (width, height) to not be aligned to macroblock boudary, it doesn't affect the bitstream decoding. And it is preferable to use 'intermediate buffer' that is macroblock aligned for decoding. In order to simplify the out-of-bound reference pixel access, the out-of-bound extrapolation rule in VC1 spec can be used to expand the expected decoded frame to the intermediate buffer dimension.
	15:8	**Reserved** Format: MBZ
	7:0	**PictureWidthInMBsMinus1 (Picture Width Minus 1 in Macroblocks)** Format: U8-1 This field indicates the width of the picture in unit of macroblocks. For example, for a 1920x1080 frame picture, PictureWidthInMBs equals 120 (1920 divided by 16).This field is used in VLD and IT modes **Value** / **Name** / **Description** 255 / Value_255 /
2	31:24	**Bitplane Buffer Pitch Minus 1** Format: U7-1 Pitch in (Bytes - 1). Specifies the bitplane buffer pitch in (#Bytes - 1). Bitplane buffer is a linear buffer. It is needed only when the bitplane is not encoded as raw, and therefore is present in the header explicitly. In VC1 Long Format (Gen6 and Gen7), it is written by an application and later read by the HW. But in VC1 Short Format (Gen7 only), it is written and read by H/W only.This field is specified for better performance **Value** / **Name** [0, FFFFFFFFh] / **Programming Notes**

MFD_VC1_LONG_PIC_STATE

		For Gen6 : The pitch must be equal to PictureWidthInMBs/2.For Gen7 VC1 Long Format : The pitch must be equal to PictureWidthInMBs/2.For Gen7 VC1 Short Format : If Pic Width is less than or equal to 2K pixels, bitplane pitch is set to 64 (one cacheline; programmed as 63) bytes per MB row. If Pic Width is greater than 2K pixels, bitplane pitch is set to 128 (two cachelines; programmed as 127) bytes per MB row.This field is not used in IT mode, used in VLD mode only.For VC1 DXVA2 Short Format, the bitplane specification is between H/W and Driver only. For Long Format, application is responsible for allocation with the driver.
	23:16	**Reserved** Format: MBZ
	15	**DmvSurfaceValid** Indicated when the DMV read surface is valid. This surface stored the direct motion vectors and Mb type.This field is set for B pictures that can refer to a previous P picture for DMV. If there is an I-picture before a B (in decoding order) then this field is not set (as a result, zero's DMV's will be assumed while decoding the B picture. That is, there is no explicit DMV buffer for an I-picture).Whne the current picture being decoded is an I, P or BI, this bit is set to 0, since there is no DMV read in these picture decoding process.This field is not used in IT mode, used in VLD mode only.
	14	**ImplicitQuantizer** Derived by driver from QUANTIZER.This field is used in intel VC1 VLD Long Format only, not used in IT and DXVA2 VC1.This bit is set to 1 when syntax element QUANTIZER=0, else its set to 0
	13	**Interpolation Rounder Contro** Used only in MC operation. This field specifies the rounding control value used in interpolation operation of motion prediction process.This field is used in VLD and IT modes. **Programming Notes** This bit field is taken from bRcontrol in DXVA_PictureParameters data structure
	12	**SyncMarker** Indicates whether sync markers are enabled/disabled. If enable, sync markers "may be" present in the current video sequence being decoded. It is a sequence level syntax element and is valid only for Simple and Main Profiles. **Value** / **Name** / **Description** 0h / Not Present / Sync Marker is not present in the bitstream 1h / Maybe present / Sync Marker maybe present in the bitstream **Programming Notes** This field is only valid in VLD mode.For Simple Profile, SyncMarker must set to 0.For Main Profile, SyncMarker can be set to 0 or 1.This field is used in both intel and MS DXVA2 VLD interface, but not used in IT mode.
	11:8	**Motion Vector Mode** This field indicates one of the following motion compensation interpolation modes for P and B pictures. The MC interpolation modes apply to prediction values of luminance blocks and are

MFD_VC1_LONG_PIC_STATE

always in quarter-sample. For chrominance blocks, it always performs bilinear interpolation with either half-pel or quarter-pel precision.Before the polarity of Chroma Half-pel or Q-pel is reversed from DXVA2 Spec, now I have fixed it to match with DXVA2 VC1 Spec.

Value	Name	Description
0XX0b		Chroma Quarter -pel + Luma bicubic. (can only be 1MV)
0XX1b		Chroma Half-pel + Luma bicubic. (can be 1MV or 4MV)
1XX0b		Chroma Quarter -pel + Luma bilinear. (can only be 1MV)
1XX1b		Chroma Half-pel + Luma bilinear

Programming Notes

Bits 11:8 are taken from bMVprecisionAndChromaRelation in DXVA_PictureParameters data structure.Bit 11 of Motion Vector Mode = 1 for Luma Bilinear MC; = 0 for Luma Bicubic MCBit 8 of Motion Vector Mode = 1 for half-sample Chroma motion = 0 for quarter-sample Chroma motion.This field is used in both VLD and IT modes.

7 **RangeReductionScale**

This field specifies whether the reference picture pixel values should be scaled up or scaled down on-the-fly, if RangeReduction is Enabled.

Value	Name	Description
0h		Scale down reference picture by factor of 2
1h		Scale up reference picture by factor of 2

Programming Notes

This bit is derived by driver for Main Profile only. Ignored in Simple and Advanced Profiles. This field is used in both VLD and IT modes.This is derived by driver from the history of RANGERED and RANGEREDFRM syntax elements (i.e. of forward/preceding reference picture) and those of the current picture. RANGERED is the same as (bPicOverflowBlocks » 3) & 1. RANGEREDFRM is the same as (bPicDeblocked » 5) & 1. For the current picture is a B picture, this field represents the state of the forward/preceding reference picture onlyDriver is responsible to keep RangeReductionScale, RangeReduction Enable and RANGERED Present Flag of current picture coherent.

6 **RangeReduction Enable**

This field specifies whether on-the-fly pixel value range reduction should be performed for the preceding (or forward) reference picture. Along with RangeReductionScale to specify whether scale up or down should be performed. It is not the same value as RANGEREDFRM Syntax Element (DXVA_PictureParameters bPicDeblocked bit 5) in the Picture Header.

Value	Name	Description
0h	Disable	Range reduction is not performed
1h	Enable	Range reduction is performed

Programming Notes

MFD_VC1_LONG_PIC_STATE

		This field is for Main Profile only. Simple Profile is always disable, and not applicable to Advanced Profile. This field is used in both VLD and IT modes.This is derived by driver from the history of RANGERED and RANGEREDFRM syntax elements (i.e. of forward/preceding reference picture) and those of the current picture.RANGERED is the same as (bPicOverflowBlocks » 3) & 1. RANGEREDFRM is the same as (bPicDeblocked » 5) & 1.For the current picture is a B picture, this field represents the state of the forward/preceding reference picture onlyDriver is responsible to keep RangeReductionScale, RangeReduction Enable and RANGERED Present Flag of current picture coherent.
	5	**LOOPFILTER Enable Flag** This filed is the decoded syntax element LOOPFILTER in bitstream. It indicates if In-loop Deblocking is ON according to picture level bitstream syntax control. This bit affects BSD unit and also the loop filter unit.When this bit is set to 1, PostDeblockOutEnable field in MFX_PIPE_MODE_SELECT command must also be set to 1. In this case, in-loop deblocking operation follows the VC1 standard - deblocking doesn't cross slice boundary.When this bit is set to 0, but PostDeblockOutEnable field in MFX_PIPE_MODE_SELECT command is set to 1. It indicates the loop filter unit is used for out-of-loop deblocking. In this case, deblocking operation does cross slice boundary.This field is used in VLD mode only, not in IT mode.
		Value / **Name** / **Description** 0h / Disable / Disables loop filter 1h / Enable / Enables loop filter
	4	**Overlap Smoothing Enable Flag** This field is the decoded syntax element OVERLAP in bitstreamIndicates if Overlap smoothing is ON at the picture levelThis field is used in both VLD and IT modes.
		Value / **Name** / **Description** 0h / Disable / to disable overlap smoothing filter 1h / Enable / to enable overlap smoothing filter
	3	**Secondfield** This flag is set for the second field in field pictures.This field is used in both VLD and IT modes.
	2:1	**Reserved** Format: MBZ
	0	**VC1 Profile** specifies the bitstream profile.This field is used in both VLD and IT modes.
		Value / **Name** / **Description** 0h / Disable / current picture is in Simple or Main Profile (No need to distinguish Simple and Main Profile) 1h / Enable / current picture is in Advanced Profile
		Programming Notes This is required because 128 is added for intra blocks post inverse transform in advanced profile and also to find out if Motion vectors are adjusted or not.

MFD_VC1_LONG_PIC_STATE

3	31	**Reserved** Format: MBZ
	30:29	**CondOver** This field is the decoded syntax element CONDOVER in a bitstream of advanced profile. It controls the overlap smoothing filter operation for an I frame or an BI frame when the picture level qualization step size PQUANT is 8 or lower.This field is used in intel VC1 VLD mode only, not in DXVA2 VC1 and IT modes. Value / Name / Description: 00b — No overlap smoothing 01b — Reserved 10b — Always perform overlap smoothing filter 11b — Overlap smoothing on a per macroblock basis based on OVERFLAGS
	28:26	**PicType (Picture Type)** This field specifies the coding type of the picture according to the Frame Coding Mode. When FCM = 00 | 01 (a Progressive or Interlaced Frame Picture):000 = I001 = P010 = B011 = BI100 = SkippedOther encodings are reservedWhen FCM = 10 | 11 (a Field Picture)000 = I/I001 = I/P010 = P/I011 = P/P100 = B/B101 = B/BI110 = BI/B111 = BI/BIAlthough, for a field picture, it is set for a field-pair, but HW will only look at one field state only, and the other field state is don't care. This field is read and qualified with the SecondField flag internally.This field is unique to intel VC1 VLD Long format, and is used in IT mode as well. For DXVA2 VC1 IT mode, driver needs to convert the DXVA2 interface to intel HW VLD Long Format interface.
	25:24	**FCM (Frame Coding Mode)** This is the same as the variable FCM defined in VC1.This field must be set to 0 for Simple and Main ProfilesThis field is unique to intel VC1 VLD Long format, and is used in IT mode as well. For DXVA2 VC1 IT mode, driver needs to convert the DXVA2 interface to intel HW VLD Long Format interface. Value / Name / Description: 00b — Disable — Progressive Frame Picture 01b — Enable — Interlaced Frame Picture 10b — Field Picture with Top Field First 11b — Field Picture with Bottom Field First
	23:21	**Reserved** Format: MBZ
	20:16	**AltPQuant (Alternative Picture Quantization Value)** This field is identical to the variable ALTPQUANT which is derived from VOPDQUANT configuration in the VC1 standard.This field must be set to 0 for Simple/Main I and BI pictures as VOPDQUANT is not present.This field is used in intel VC1 VLD Long Format mode only, not used in DXVA2 VC1 VLD and IT modes.
	15:13	**Reserved** Format: MBZ

MFD_VC1_LONG_PIC_STATE

	12:8	**PQuant (Picture Quantization Value)** Format: U5 This is the same as the calculated variable PQUANT in VC1 standard where PQuant = PQINDEX, except when QUANTIZER = 0 and PQINDEX > 8, it is given asPQuant = (PQINDEX < 29) ? PQINDEX - 3 : PQINDEX*2 - 31This field is used in all picture types (I, P, B and BI) and all operating modes (IT mode and intel and DXVA2 VLD modes).
	7:0	**BScaleFactor** BScaleFactorThis field is the scale factor for computing Direct-mode motion vectors. It is derived from the variable BFRACTION in the VC1 standard, section 8.4.5.4.There are only 21 valid values corresponding to the 21 encodings of BFRACTION as shown in the table here. Other values are reserved.MSB of this field can be used to determine if BFRACTION is greater than or equal to 1/2, which is used to determine Motion Prediction Type for B pictures. Effectively, condition "BFRACTION >= 1/2" is equivalent to condition "BScaleFactor >= 128".This field is only valid for B pictures. This field is used only in intel VC1 VLD Long format mode, it is not used in DXVA2 VC1 VLD and IT modes.BFRACTION VLCBFRACTIONBScaleFactor0001/21280011/3850102/31700111/4641003/41921011/5511102/510211100003/515311100014/520411100101/64311100115/621511101001/73711101012/77411101103/711111101114/714811110005/718511110016/722211110101/83211110113/89611111005/816011111017/8224
4	31:30	**Reserved** Format: MBZ
	29:28	**UnifiedMvMode (Unified Motion Vector Mode)** This field is a combination of the variables MVMODE and MVMODE2 in the VC1 standard, for parsing Luma MVD from the bitstream. This field is used to signal 1MV vs 4MVallowed (Mixed Mode). This field is also used to signal Q-pel or Half-pel MVD read from the bitstream. The bicubic or bilinear Luma MC interpolation mode is duplicate information from Motion Vector Mode field, and is ignored here.This field is used in intel VC1 VLD Long Format mode only, it is not used in DXVA2 VC1 VLD and IT modes.

Value	Name	Description
00b		Mixed MV, Q-pel bicubic
01b		1-MV, Q-pel bicubic
10b		1-MV half-pel bicubic
11b		1-MV half-pel bilinear

	27	**FourMvSwitch (Four Motion Vector Switch)** This field indicates if 4-MV is present for an interlaced frame P picture. It is identical to the variable 4MVSWITCH (4 Motion Vector Switch) in VC1 standard.This field is used in intel VC1 VLD Long Format mode only, it is not used in DXVA2 VC1 VLD and IT modes.

Value	Name	Description
0h	Disable	only 1-MV
1h	Enable	1, 2, or 4 MVs

MFD_VC1_LONG_PIC_STATE

	26	**FastUVMCFlag (Fast UV Motion Compensation Flag)** This field specifies whether the motion vectors for UV is rounded to half or full pel position. It is identical to the variable FASTUVMC in VC1 standard.This field is used in both VLD and IT modes.It is derived from FASTUVMC = (bPicSpatialResid8 » 4) & 1 in both VLD and IT modes, and should have the same value as Motion Vector Mode LSBit.

Value	Name	Description
0h		no rounding
1h		quarter-pel offsets to half/full pel positions

	25	**RefFieldPicPolarity (Reference Field Picture Polarity)** This field specifies the polarity of the one reference field picture used for a field P picture. It is derived from the variable REFFIELD defined in VC1 standard and is only valid when one field is referenced (NUMREF = 0) for a field P picture.When NUMREF = 0 and REFFIELD = 0, this field is the polarity of the reference I/P field that is temporally closest; When NUMREF = 0 and REFFIELD = 1, this field is the polarity of the reference I/P field that is the second most temporally closest. The distance is measured based on display order but ignoring the repeated field if present (due to RFF = 1).This field is unique to intel VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.

Value	Name	Description
0h		Top (even) field
1h		Bottom (odd) field

	24	**NumRef (Number of References)** This field indicates how many reference fields are referenced by the current (field) picture. It is identical to the variable NUMREF in the VC1 standard. This field is only valid for field P picture (FCM = 10 | 11).This field is unique to intel VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.

Value	Name	Description
0h		One field referenced
1h		Two fields referenced

	23:20	**BwdRefDist (Reference Distance)** This field is valid only in B field pictures giving the value of BRFD. The field is ignored in P Picture.This field is unique to intel VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.
	19:16	**FwdRefDist (Reference Distance)**

Format:	U4

This field is the number of frames between the current frame and its reference frame. It is derived from the syntax element REFDIST (P Reference Distance) in the VC1 standard. 0 means that the previous frame is the reference frame.It has the same value as of FRFD for both P and B field pictures.This field is unique to intel VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.

Value	Name
[0, 15]	

MFD_VC1_LONG_PIC_STATE

15:12 **Reserved**

Format:	MBZ

11:10 **ExtendedDMVRange (Extended Differential Motion Vector Range Flag)**

This field specifies the differential motion vector range in interlaced pictures. It is equivalent to the variable DMVRANGE in the VC1 standard. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.

Value	Name	Description
00b		No extended range
01b		Extended horizontally
10b		Extended vertically
11b		Extended in both directions

9:8 **ExtendedMVRange (Extended Motion Vector Range Flag)**

This field specifies the motion vector range in quarter-pel or half-pel modes. It is equivalent to the variable MVRANGE in the VC1 standard. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes

Value	Name	Description
00b		[-256, 255] x [-128, 127]
01b		512, 511] x [-256, 255]
10b		[-2048, 2047] x [-1024, 1023]
11b		[-4096, 4095] x [-2048, 2047]

7:4 **AltPQuantEdgeMask (Alternative Picture Quantization Edge Mask)**

This field is a bit mask for the four edges in clock-wise order, indicating whether AltPQuant is used for the edge macroblocks.It is derived based on the following variables DQUANT, DQUANTFRM, DQPROFILE, DQSBEDGE, DQDBEDGE, and DQBILEVEL defined in the VC1 standard, as shown in Error! Reference source not found..This field is valid only if AltPQuantConfig is 01. Bit 0: Left picture edge macroblocksBit 1: Top picture edge macroblocksBit 2: Right picture edge macroblocksBit 3: Bottom picture edge macroblocksThis field is unique to intel VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.

3:2 **AltPQuantConfig (Alternative Picture Quantization Configuration)**

This field specifies the way AltPQuant is used in the picture. It determines how to compute the macroblock quantizer step size, MQUANT. It is derived based on the following variables DQUANT, DQUANTFRM, DQPROFILE, DQSBEDGE, DQDBEDGE, and DQBILEVEL defined in the VC1 standard, as shown in Error! Reference source not found..This field is unique to intel VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.

Value	Name	Description
00b		AltPQuant not used
01b		AltPQuant is used and applied to edge macroblocks only
10b		MQUANT is encoded in macroblock layer
11b		AltPQuant and PQuant are selected on macroblock basis

MFD_VC1_LONG_PIC_STATE

	1	**HalfQP** This field is used for inverse quantization of AC coefficients. It is valid only when PQuant is used.This field is unique to intel VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.
	0	**PQuantUniform** Indicating if uniform quantization applies to the picture. It is used for inverse quantization of the AC coefficients.QUANTIZER 001123PQUANTIZER - -01--PQINDEX>=9<=8----PQuantUniform010201ImplicitQuantizer = 0, and PQuantUniform = 0 is used to represent 2 cases : 1) QUANTIZER=01 and PQUANTIZER=0; and 2) QUANTIZER = 10b.ImplicitQuantizer = 0, and PQuantUniform = 1 is used to represent 2 cases : 1) QUANTIZER=01 and PQUANTIZER=1; and 2) QUANTIZER = 11bThis field is unique to intel VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.

Value	Name	Description
0h		Non-uniform
1h		Uniform

5	31	**BitplanePresentFlag (Bitplane Buffer Present Flag)** This field indicates whether the bitplane buffer is present for the picture. If set, at least one of the fields listed in bits 22:16 is coded in non-raw mode, and Bitplane Buffer Base Address field in the VC1_BSD_BUF_BASE_STATE command points to the bitplane buffer. Otherwise, all the fields that are applicable for the current picture in bits 22:16 must be coded in raw mode.This field is unique to intel VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.

Value	Name	Description
0h		bitplane buffer is not present
1h		bitplane buffer is present

	30	**ForwardMbRaw** This field indicates whether the FORWARDMB field is coded in raw or non-raw mode.This field is only valid when PictureType is B.This field is unique to intel VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.

Value	Name	Description
0h		non-raw mode
1h		raw mode

	29	**MvTypeMbRaw** This field indicates whether the MVTYPREMB field is coded in raw or non-raw mode.This field is only valid when PictureType is P.This field is unique to intel VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.

Value	Name	Description
0h		Non-Raw Mode
1h		Raw Mode

MFD_VC1_LONG_PIC_STATE

28 **SkipMbRaw**

This field indicates whether the SKIPMB field is coded in raw or non-raw mode.This field is only valid when PictureType is P or B.0 = non-raw mode1 = raw modeThis field is unique to intel VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.

Value	Name	Description
0h	Disable	Non-Raw Mode
1h	Enable	Raw Mode

27 **DirectMbRaw**

This field indicates whether the DIRECTMB field is coded in raw or non-raw mode.This field is only valid when PictureType is P or B.This field is unique to intel VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.

Value	Name	Description
0h		Non-Raw Mode
1h		Raw Mode

26 **OverflagsRaw**

This field indicates whether the OVERFLAGS field is coded in raw or non-raw mode.This field is only valid when PictureType is I or BI.This field is unique to intel VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.

Value	Name	Description
0h		Non-Raw Mode
1h		Raw Mode

25 **AcPredRaw**

This field indicates whether the ACPRED field is coded in raw or non-raw mode.This field is only valid when PictureType is I or BI.This field is unique to intel VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.

Value	Name	Description
0h	Disable	Non-Raw Mode
1h	Enable	Raw Mode

24 **FieldTxRaw**

This field indicates whether the FIELDTX field is coded in raw or non-raw mode.This field is only valid when PictureType is I or BI.This field is unique to intel VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.

Value	Name	Description
0h	Disable	Non-Raw Mode
1h	Enable	Raw Mode

23 **Reserved**

Format:	MBZ

MFD_VC1_LONG_PIC_STATE

	22:20	**MvTab (Motion Vector Table)** Format: U3 This field specifies which motion vector table(s) is (are) used for motion vector (differential) decoding in a P or B picture. This field is the combination of the variables MVTAB and IMVTAB in the VC1 standard. Two bits are defined for progressive frame pictures; And two or three bits are defined for interlaced field/frame pictures depending on NUMREF and P or B picture types.This field is valid for P and B pictures. It is not valid for I pictures.For P or B progressive frame pictures0 = Motion Vector Differential VLD Table 01 = Motion Vector Differential VLD Table 12 = Motion Vector Differential VLD Table 23 = Motion Vector Differential VLD Table 3The other encodings are reservedFor P interlace field pictures with NUMREF = 0 or P/B interlace frame pictures0 = 1-Reference Table 01 = 1-Reference Table 12 = 1-Reference Table 23 = 1-Reference Table 3The other encodings are reservedFor P interlace field picture with NUMREF = 1 or B interlaced field pictures0 = 2-Reference Table 01 = 2-Reference Table 12 = 2-Reference Table 23 = 2-Reference Table 34 = 2-Reference Table 45 = 2-Reference Table 56 = 2-Reference Table 67 = 2-Reference Table 7The other encodings are reservedThis field is unique to intel VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.
	19:18	**FourMvBpTab (4-MV Block Pattern Table)** This field specifies which table is used to decode the 4-MV block pattern (4MVBP) syntax element in 4-MV macroblocks. It is identical to the variables 4MVBPTAB in the VC1 standard, section 9.1.1.37. This field is valid only in interlace frame P, B pictures, or interlace field P, B pictures. It is not valid for I picture.For interlace field P and B pictures, it is only valid if UnifiedMvMode is equal to Mixed-MV Type. For interlace frame P picture, it is only valid if FourMvSwitch is 1.For interlace frame B picture, it is always valid.0 = 4MVBP Table 01 = 4MVBP Table 12 = 4MVBP Table 23 = 4MVBP Table 3This field is unique to intel VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.
	17:16	**TwoMvBpTab (2MV Block Pattern Table)** This field specifies which table is used to decode the 2MV block pattern (2MVBP) syntax element in 2MV field macroblocks. It is identical to the variables 2MVBPTAB in the VC1 standard, section 9.1.1.36. This field is valid only in interlace frame P/B pictures. It is not valid for I picture, nor for interlace field P or B pictures.0 = 2MVBP Table 01 = 2MVBP Table 12 = 2MVBP Table 23 = 2MVBP Table 3This field is unique to intel VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.
	15:14	**Reserved** Format: MBZ
	13:12	**TransType (Picture-level Transform Type)** Format: U2 This field specifies the Transform Type at picture level. It is identical to the variable TTFRM in the VC1 standard, section 7.1.1.41.This field is only valid when TransTypeMbFlag is 1. Otherwise, it is reserved and MBZ.This field is set to 00 when VSTRANSFORM is 0 in the entry point layer.00 = 8x8 Transform01 = 8x4 Transform10 = 4x8 Transform11 = 4x4 TransformThis field is unique to intel VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.

MFD_VC1_LONG_PIC_STATE

	11	**TransTypeMbFlag (Macroblock Transform Type Flag)** This field indicates whether Transform Type is fixed at picture level or variable at macroblock level. It is identical to the variable TTMBF in the VC1 standard, section 7.1.1.40.This field is set to 1 when VSTRANSFORM is 0 in the entry point layer.This field is unique to intel VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.

Value	Name	Description
0h		variable transform type in macroblock layer
1h		use picture level transform type TransType

	10:8	**MbModeTab (Macroblock Mode Table)** This field signals which code table is used to decode the macroblock mode syntax element (MBMODE) in the macroblock layer in a P or B picture. This field is identical to the variables MBMODETAB in the VC1 standard, section 9.1.1.33. This field is valid for interlace frame P, B picture and interlace field P, B picture. It is not valid for I picture, nor progressive frame P, B pictures.Two bits are defined for interlace frame P, B pictures; And three bits are defined for interlaced field P, B pictures.Two bits are defined for interlace frame P, B pictures. There are two set of code tables selected based on if UnifiedMvMode is equal to 4-MV Type or not. 0 = Code Table 01 = Code Table 12 = Code Table 23 = Code Table 3Other encodings are invalidThree bits are defined for interlace field P, B pictures. There are two set of code tables selected based on if UnifiedMvMode is equal to Mixed-MV Type or not. 0 = Code Table 01 = Code Table 12 = Code Table 23 = Code Table 34 = Code Table 45 = Code Table 56 = Code Table 67 = Code Table 7This field is unique to intel VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.
	7:6	**TransAcY (Picture-level Transform Luma AC Coding Set Index, TRANSACTABLE2** BitFieldDesc
	5:4	**TransAcUV (Picture-level Transform Chroma AC Coding Set Index, TRANSACTABLE)** This field, together with PQINDEX, specifies which intra AC coding set to be used for decoding the non-zero AC coefficients in a coded luma (Y) block. This field is the combination of the variables TRANSACFRM and TRANSACFRM2 in the VC1 standard.For I pictures, TransAcY is the same as TRANSACFRM2. For other pictures, it is the same as TRANSACFRM, and therefore must be programmed to be the same as TransAcUV. This field is valid for all picture types.0 = Coding set index 01 = Coding set index 12 = Coding set index 23 is invalidThis field is unique to intel VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.
	3	**TransDcTab (Intra Transform DC Table)** This field specifies whether the low motion tables or the high motion tables are used to decode the Transform DC coefficients in intra-coded blocks. This field is identical to the variable TRANSDCTAB in the VC1 standard, section 8.1.1.2.This field is valid for all picture types.This field is unique to intel VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.

Value	Name	Description
0h		The high motion tables
1h		The low motion tables

MFD_VC1_LONG_PIC_STATE		
	2:0	**CbpTab (Coded Block Pattern Table)** This field specifies the table used to decode the CBPCY syntax element for each coded macroblock in P and B pictures. This field is combination of the variable CBPTAB for P and B frame pictures and the variable ICBPTAB in interlace field P, B pictures and interlace frame P, B pictures in the VC1 standard (Table 52 and Table 102).This field is reserved and MBZ for I or BI pictures as I only has a fixed table.000 = Table 0 (Table 169 for P, B frames or Table 124 otherwise)001 = Table 1 (Table 170 for P, B frames or Table 125 otherwise)010 = Table 2 (Table 171 for P, B frames or Table 126 otherwise)011 = Table 3 (Table 172 for P, B frames or Table 127 otherwise)100 = Table 4 (Table 128 for interlace field/frame P, B pictures)101 = Table 5 (Table 129 for interlace field/frame P, B pictures)110 = Table 6 (Table 130 for interlace field/frame P, B pictures)111 = Table 7 (Table 131 for interlace field/frame P, B pictures)This field is unique to intel VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.

MFD_VC1_SHORT_PIC_STATE

MFD_VC1_SHORT_PIC_STATE

Source: VideoCS
Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFD_VC1_SHORT_PIC_STATE Format: OpCode
	26:24	**Media Command Opcode** Default Value: 2h VC1_DEC Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h Format: OpCode
	20:16	**SubOpcode B** Default Value: 0h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 0003h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:24	**Reserved** Format: MBZ
	23:16	**Picture Height** Format: U8-1 Picture Height in Macroblocks This field indicates the height of the picture in unit of macroblocks. For example, for a 1920x1080 frame picture, PictureHeightInMBs equals 68 (1080 divided by 16, and rounded up, i.e. effectively specified as 1088 instead).This field is used in VLD and IT modes.Note: Even though the Advanced Profile allows frame dimensions (width, height) to not be aligned to macroblock boudary, it doesn't affect the bitstream decoding. And it is preferable to use 'intermediate buffer' that is macroblock aligned for decoding. In order to simplify the out-of-bound reference pixel access, the out-of-bound extrapolation rule in VC1 spec can be used to expand the expected decoded frame to the intermediate buffer dimension.

MFD_VC1_SHORT_PIC_STATE

		Value / **Name** / **Description** [0,127] / Value_0_to_127 / [1, 128] MB [128,255] / Value_128_to_255 /
	15:8	**Reserved** Format: MBZ
	7:0	**Picture Width** Format: U8-1 Picture Width in Macroblocks This field indicates the width of the picture in unit of macroblocks. For example, for a 1920x1080 frame picture, PictureWidthInMBs equals 120 (1920 divided by 16).This field is used in VLD and IT modes. **Value** / **Name** / **Description** [0,127] / Value_0_to_127 / [1, 128] MB [128,255] / Value_128_to_255 /
2	31:24	**Bitplane Buffer Pitch Minus 1** Format: U7-1 Pitch in Bytes Specifies the bitplane buffer pitch in (#Bytes - 1). Bitplane buffer is a linear buffer. It is needed only when the bitplane is not encoded as raw, and therefore is present in the header explicitly. In VC1 Long Format (Gen6 and Gen7), it is written by an application and later read by the HW. In VC1 Long Format (Gen6 and Gen7), it is written by an application, and later read by the HW. But in VC1 Short Format (Gen7 only), it is written and read by H/W only.This field is specified for better performanceFor Gen6 : The pitch must be equal to PictureWidthInMBs/2.For Gen7 VC1 Long Format : The pitch must be equal to PictureWidthInMBs/2.For Gen7 VC1 Short Format : If Pic Width is less than or equal to 2K pixels, bitplane pitch is set to 64 (one cacheline; programmed as 63) bytes per MB row. If Pic Width is greater than 2K pixels, bitplane pitch is set to 128 (two cachelines; programmed as 127) bytes per MB row.This field is not used in IT mode, used in VLD mode only.For VC1 DXVA2 Short Format, the bitplane specification is between H/W and Driver only. For Long Format, application is responsible for allocation with the driver.
	23	**Interpolation Rounder Control** Used only in MC operation. This field specifies the rounding control value used in interpolation operation of motion prediction process. Note: This bit field is taken from bRcontrol in DXVA_PictureParameters data structure This field is used in VLD and IT modes.
	22:20	**Reserved** Format: MBZ
	19:16	**Motion Vector Mode** This field indicates one of the following motion compensation interpolation modes for P and B pictures. The MC interpolation modes apply to prediction values of luminance blocks and are always in quarter-sample. For chrominance blocks, it always performs bilinear interpolation with either half-pel or quarter-pel precision.0XX0 = Chroma Quarter -pel + Luma bicubic. (can only be 1MV)0XX1 = Chroma Half-pel + Luma bicubic. (can be 1MV or 4MV)1XX0 = Chroma Quarter -pel + Luma bilinear. (can only be 1MV)1XX1 = Chroma Half-pel + Luma bilinearNote: Bits 19:16 are

MFD_VC1_SHORT_PIC_STATE

		taken from bMVprecisionAndChromaRelation in DXVA_PictureParameters data structure.Bit 19 of Motion Vector Mode = 1 for Luma Bilinear MC; = 0 for Luma Bicubic MCBit 16 of Motion Vector Mode = 1 for half-sample Chroma motion = 0 for quarter-sample Chroma motion.This field is used in both VLD and IT modes.Before the polarity of Chroma Half-pel or Q-pel is reversed from DXVA2 Spec, now I have fixed it to match with DXVA2 VC1 Spec. ???
	15	**DmvSurfaceValid** Indicated when the DMV read surface is valid. This surface stored the direct motion vectors. This field is set fo B pictures that can refer to a previous P picture for DMV. If there is an I-picture before a B (in decoding order) then this field is not set (as a result, zero's DMV's will be assumed while decoding the B picture. That is, there is no explicit DMV buffer for an I-picture). This field is not used in IT mode, used in VLD mode only.
	14:12	**Reserved** Format: MBZ
	11	**VC1 Profile** specifies the bitstream profile. Note: This is required because 128 is added for intra blocks post inverse transform in advanced profile and also to find out if Motion vectors are adjusted or not. This field is used in both VLD and IT modes.
	10:6	**Reserved** Format: MBZ
	5	**Backward Prediction Present Flag** Note : a B picture that only uses forward prediction may have this flag set to 1 as well. Driver may still need to provide a valid reference picture index. This field is used in both DXVA2 VC1 VLD mode and IT mode. It is the same parameter as bPicBackwardPrediction in DXVA2 VC1 spec. The Intra Picture Flag, Backward Prediction Present Flag and RefPicFlag are used to derive the picture type, as specified in PTYPE for a frame, and in FPTYPE for a field, in DXVA2 VC1 VLD and IT mode.
	4	**Intra Picture Flag** This field is used in both DXVA2 VC1 VLD mode and IT mode. It is the same parameter as bPicIntra in DXVA2 VC1 spec. The Intra Picture Flag, Backward Prediction Present Flag and RefPicFlag are used to derive the picture type, as specified in PTYPE for a frame, and in FPTYPE for a field, in DXVA2 VC1 VLD and IT mode.

Value	Name	Description
0h	**[Default]**	current picture is in Simple or Main Profile (No need to distinguish Simple and Main Profile)
1h		current picture is in Advanced Profile

Value	Name	Description
0h		entire picture can have a mixture of intra and inter MB type or just inter MB type.

MFD_VC1_SHORT_PIC_STATE

		1h | | entire picture is coded in intra MB type
	3	**SecondField** This flag is set for the second field in field pictures.This field is used in both VLD and IT modes.
	2	**Reserved** Format: MBZ
	1:0	**Picture Structure** This field is used in both DXVA2 VC1 VLD mode and IT mode. It is the same parameter as bPicStructure in DXVA2 VC1 spec. The Picture Structure and Progressive Pic Type are used to derive the picture structure as specified in FCM, in DXVA2 VC1 VLD and IT mode.

Value	Name	Description
01b		top field (bit 0)
10b		bottom field (bit 1)
11b		frame (both fields are present)
00b		illegal

3	31	**Reserved** Format: MBZ
	30	**Overlap Smoothing Enable Flag** This field is the decoded syntax element OVERLAP in bitstreamIndicates if Overlap smoothing is ON at the picture levelThis field is used in both VLD and IT modes

Value	Name	Description
0h	Disable	to disable overlap smoothing filter
1h	Enable	to enable overlap smoothing filter

	29	**Range Reduction Scale** Access: None This field specifies whether the reference picture pixel values should be scaled up or scaled down on-the-fly, if RangeReduction is Enabled.NOTE: This bit is derived by driver for Main Profile only. Ignored in Simple and Advanced Profiles. This field is used in both VLD and IT modes.This is derived by driver from the history of RANGERED and RANGEREDFRM syntax elements (i.e. of forward/preceding reference picture) and those of the current picture. RANGERED is the same as (bPicOverflowBlocks » 3) & 1. RANGEREDFRM is the same as (bPicDeblocked » 5) & 1. For the current picture is a B picture, this field represents the state of the forward/preceding reference picture onlyDriver is responsible to keep RangeReductionScale, RangeReduction Enable and RANGERED Present Flag of current picture coherent.

Value	Name	Description
0h	Disable **[Default]**	Scale down reference picture by factor of 2
1h	Enable	Scale up reference picture by factor of 2

MFD_VC1_SHORT_PIC_STATE

	28	**Range Reduction Enable** This field specifies whether on-the-fly pixel value range reduction should be performed for the preceding (or forward) reference picture. Along with RangeReductionScale to specify whether scale up or down should be performed. It is not the same value as RANGEREDFRM Syntax Element (DXVA_PictureParameters bPicDeblocked bit 5) in the Picture Header.This field is for Main Profile only. Simple Profile is always disable, and not applicable to Advanced Profile. This field is used in both VLD and IT modes.This is derived by driver from the history of RANGERED and RANGEREDFRM syntax elements (i.e. of forward/preceding reference picture) and those of the current picture.RANGERED is the same as (bPicOverflowBlocks » 3) & 1. RANGEREDFRM is the same as (bPicDeblocked » 5) & 1.For the current picture is a B picture, this field represents the state of the forward/preceding reference picture onlyDriver is responsible to keep RangeReductionScale, RangeReduction Enable and RANGERED Present Flag of current picture coherent.

Value	Name	Description
0h	Disable **[Default]**	Range reduction is not performed
1h	Enable	Range reduction is performed

	27:24	**Reserved**

Format:	MBZ

	23:22	**Progressive Pic Type** This field is used in both DXVA2 VC1 VLD mode and IT mode. It is the same parameter as bPicExtrapolation in DXVA2 VC1 spec.The Picture Structure and Progressive Pic Type are used to derive the picture structure as specified in FCM, in DXVA2 VC1 VLD and IT mode.

Value	Name	Description
0		progressive only picture
1		progressive only picture
2		interlace picture (frame-interlace or field-interlace)
3		illegal

	21	**Reserved**

Format:	MBZ

	20:16	**P-Pic Ref Distance**

Access:	None

This element defines the number of frames between the current frame and the reference frame. It is the same as the REFDIST SE in VC1 interlaced field picture header. It is present if the entry-level flag REFDIST_FLAG == 1, and if the picture type is not one of the following types: B/B, B/BI, BI/B, BI/BI. If the entry level flag REFDIST_FLAG == 0, REFDIST shall be set to the default value of 0.This field is used in DXVA2 VC1 VLD mode only, not used in IT and intel VC1 VLD Long Format modes.

Value	Name
0-16	unsigned integer
0h	**[Default]**

MFD_VC1_SHORT_PIC_STATE

15:14 QUANTIZER

Value	Name	Description
00b		implicit quantizer at frame leve
01b		explicit quantizer at frame level, and use PQUANTIZER SE to specify uniform or non-uniform
10b		explicit quantizer, and non-uniform quantizer for all frames
11b		explicit quantizer, and uniform quantizer for all frames

13 MULTIRES Present Flag (for Simple/Main Profile only)

Value	Name	Description
0h		RESPIC Parameter is present in the picture header
1h		RESPIC Parameter is present in the picture header

12 SYNCMARKER Present Flag (for Simple/Main Profile only)

Value	Name	Description
0		Bitstream for Simple and Main Profile has no sync marker
1		Bitstream for Simple and Main Profile may have sync marker(s)

11 RANGERED Present Flag (for Simple/Main Profile only)

It is needed for Picture Header Parsing.Driver is responsible to keep RangeReductionScale, RangeReduction Enable and RANGERED Present Flag of current picture coherent.

Value	Name	Description
0		Range Reduction Parameter (RANGEREDFRM) is not present in the picture header
1		Range Reduction Parameter (RANGEREDFRM) is present in the picture header.

10:8 MAXBFRAMES

Number of consecutive B Frames.

7 PANSCAN Present Flag

Value	Name	Description
0		Pan Scan Parameters are not present in the picture header
1		Pan Scan Parameters are present in the picture header

6 REFDIST_FLAG

For header parsing REFDIST.This is used in DXVA2 VC1 VLD mode only, not used in IT and intel VC1 VLD modes.

5 LOOPFILTER Enable Flag

This filed is the decoded syntax element LOOPFILTER in bitstream. It indicates if In-loop Deblocking is ON according to picture level bitstream syntax control. This bit affects BSD unit and also the loop filter unit.When this bit is set to 1, PostDeblockOutEnable field in MFX_PIPE_MODE_SELECT command must also be set to 1. In this case, in-loop deblocking operation follows the VC1 standard - deblocking doesn't cross slice boundary.When this bit is set to 0, but PostDeblockOutEnable field in MFX_PIPE_MODE_SELECT command is set to 1. It

MFD_VC1_SHORT_PIC_STATE

indicates the loop filter unit is used for out-of-loop deblocking. In this case, deblocking operation does cross slice boundary.This field is used in VLD mode only, not in IT mode.

Value	Name	Description
0		In-Loop-Deblocking-Filter is disabled
1		In-Loop-Deblocking-Filter is enabled

4 — FastUVMCFlag (Fast UV Motion Compensation Flag)

This field specifies whether the motion vectors for UV is rounded to half or full pel position. It is identical to the variable FASTUVMC in VC1 standard.This field is used in both VLD and IT modes.It is derived from FASTUVMC = (bPicSpatialResid8 » 4) & 1 in both VLD and IT modes, and should have the same value as Motion Vector Mode LSBit.

Value	Name	Description
0h		no rounding
1h		quarter-pel offsets to half/full pel positions

3 — EXTENDED_MV Present Flag

BitFieldDesc

Value	Name	Description
0h		Extended_MV is not present in the picture header
1h		Extended_MV is present in the picture header

2:1 — DQUANT

Access:	None
Format:	U2

Use for Picture Header Parsing of VOPDUANT elements

Value	Name	
0h	**[Default]**	
00b		no VOPDQUANT elements; Quantizer cannot vary in frame, same quantization step size PQUANT is used for all MBs in the frame
01b		refer to VC1 Spec. for all the MB position dependent quantizer selection
10b		The macroblocks located on the picture edge boundary shall be quantized with ALTPQUANT while the rest of the macroblocks shall be quantized with PQUANT.
11b	Reserved	

0 — VSTRANSFORM flag

Value	Name	Description
0h	Disable	variable-sized transform coding is not enabled
1h	Enable	variable-sized transform coding is enabled

4 — 31:29 — Reserved

Format:	MBZ (for possible future change to BFraction Enumeration)

MFD_VC1_SHORT_PIC_STATE

	28:24	**BFraction Enumeration** This field is the scale factor for computing Direct-mode motion vectors. It is derived from the variable BFRACTION in the VC1 standard, section 8.4.5.4.There are only 21 valid values corresponding to the 21 encodings of BFRACTION as shown in the table here. Other values are reserved. The VLD decoded value of BFRACTION (from the picture header) is mapped into an enum value from 0 to 20.(??? MSB of this field can be used to determine if BFRACTION is greater than or equal to 1/2, which is used to determine Motion Prediction Type for B pictures. Effectively, condition "BFRACTION >= 1/2" is equivalent to condition "ScaleFactor >= 128". ??? How can the enum replicate this feature ???)This field is only valid for B pictures. This field is used only in DXVA2 VC1 VLD mode, it is not used in Intel VC1 VLD Long Format mode and IT mode.BFRACTION VLCBFRACTION Enum0001/200011/310102/320111/431003/441011/551102/5611100003/5711100014/5811100101/6911100115/61011101001/71111101012/71211101103/71311101114/71411110005/71511110016/71611110101/81711110113/81811111005/81911111017/8201111111BI Pic Indicator31 (optional)
	23	**Reserved** Format: MBZ Advanced Profile only; RANGE_MAPY_FLAG Range Mapping not supported
	22:20	**Reserved** Format: MBZ Advanced Profile only; RANGE_MAPY Range Mapping not supported
	19	**Reserved** Format: MBZ Advanced Profile only; RANGE_MAPUV_FLAG Range Mapping not supported
	18:16	**Reserved** Format: MBZ Advanced Profile only; RANGE_MAPUV Range Mapping not supported
	15:9	**Reserved** Format: MBZ
	8	**4MV Allowed Flag**
	7	**POSTPROC Flag**
	6	**PULLDOWN**
	5	**INTERLACE**
	4	**TFCNTRFLAG**
	3	**FINTERFLAG**
	2	**REFPIC Flag** For a BI picture, REFPIC flag must set to 0For I and P picture, REFPIC flag must set to 0.For a B picture, REFPIC flag must set to 0, except for a B-field in interlaced field mode which can be 0 or 1 (e.g. the top B field can be used as a reference for decoding its corresponding bottom B-field in a field pair).In VLD mode, this flag cannot be used as an optimization signaling for an I or P picture that is not used as a reference picture.This field is used in both DXVA2 VC1 VLD mode and IT mode. It is the same parameter as bPicDeblockConfined[bit2] in DXVA2 VC1 spec.The Intra Picture Flag, Backward Prediction Present Flag and RefPicFlag are used to derive the picture type, as specified in PTYPE for a frame, and in FPTYPE for a field, in DXVA2 VC1 VLD and IT mode.

MFD_VC1_SHORT_PIC_STATE

		Value / **Name** / **Description** 0h / / the current picture after decoded, will never used as a reference picture 1h / / the current picture after decoded, will be used as a reference picture later
	1	**PSF**
	0	**EXTENDED_DMV Present Flag** **Value** / **Name** / **Description** 0h / **[Default]** / Extended_DMV is not present in the picture header 1h / / Extended_DMV is present in the picture header

MFX_AVC_IMG_STATE

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** Format: MBZ
	11:0	**DWord Length** Default Value: 0Ch Excludes DWord (0,1) 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** Format: MBZ

MFX_AVC_IMG_STATE

	15:0	**Frame Size** Format: U16 in MB unit This field is only valid for encoder. Its ignored for decoder.The value for FrameSizeInMBs must match the product of FrameWidthInMBs and FrameHeightInMBs. For encode mode if frame size is 4Kx4K, then we need to specify 65535 in this field as we dont have enough bits to specify 65536. This is used only for cabac zero work insertion when in encode mode. Value: [1,65535] — Name: — Description: representing Number of MBs [1,65535]
2	31:24	**Reserved** Format: MBZ (bit[31:24] must be zero to match the DXVA 16-bit definition for FrameHeightInMBsMinus1)
	23:16	**Frame Height** 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: [0,255] — Name: — Description: representing height [1,256]
	15:8	**Reserved** Format: MBZ (bit[15:8] must be zero to match the DXVA 16-bit definition for FrameWidthInMBsMinus1)
	7:0	**Frame Width** 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

MFX_AVC_IMG_STATE

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

Format:	MBZ

(bit[31:29] must be zero to match the DXVA2 8-bit definition for InitQpChroma[1])

28:24 — Second Chroma QP Offset

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

Format:	MBZ

(bit[23:21] must be zero to match the DXVA2 8-bit definition for InitQpChroma[1])

20:16 — First Chroma QP Offset

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

15:14 — Reserved

Format:	MBZ

13 — RhoDomain Rate Control Enable

Format:	Enable

This field indicates if RhoDomain related parameters are present in the MFX_AVC_IMAGE_STATE. (AverageMacroblockQP). It enables the Rho Domain statistics collection.

Value	Name	Description
0	Disable	RhoDomain rate control parameters are not present in MFX_AVC_IMAGE_STATE
1	Enable	RhoDomain rate control parameters are present in MFX_AVC_IMAGE_STATE.

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

12 — Weighted_Pred_Flag

Format:	Enable

(This field is defined differently from Gen6, Gen7 follows strictly DXVA2 AVC interface.)

MFX_AVC_IMG_STATE

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

(This field is defined differently from DevSNB; DevIVB follows strictly DXVA2 AVC interface.)

Value	Name	Description
0	DEFAULT **[Default]**	Specifies that the default weighted prediction is used for B slices
1	EXPLICIT	Specifies that explicit weighted prediction is used for B slices
2	IMPLICIT	Specifies that implicit weighted prediction is used for B slices.
3	Reserved	Illegal value

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

9:8 **ImgStruct - Image Structure, img_structure[1:0]**

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**

Format:	MBZ

MFX_AVC_IMG_STATE

4 — 31:16 — MinFrameWSize

Default Value:	0h
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

Format:	Enable

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

14 — LoadSlicePointerFlag

Format:	Enable

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
0	Disable	Load BitStream Pointer only once for the first slice of a frame
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

13 — Reserved

12 — MvUnpackedFlag

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

Value	Name	Description

MFX_AVC_IMG_STATE

0	PACKED	use packed MV format (compliant to DXVA)
1	UNPACKED	use unpacked 8MV/32MV format only

11:10 ChromaFormatIdc

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

Value	Name	Description
00b	monochrome picture	Desc
01b	4:2:0 picture	Desc
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

Format:	MBZ

8 MbMvFormatFlag

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

Value	Name	Description
0	IGNORE	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.
1	FOLLOW	HW PAK will follow 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

Entropy Coding Flag, entropy_coding_flag

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

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.

MFX_AVC_IMG_STATE

6 **ImgDisposableFlag**

Current Img Disposable Flag or Non-Reference Picture Flag

Value	Name	Description
0	REFERENCE	the current decoding picture may be used as a reference picture for others
1	DISPOSABLE	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)

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**

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

Value	Name	Description
0	INTRA_AND_INTER	allows both intra and inter neighboring MB to be used in the intra-prediction encoding of the current MB.
1	INTRA_ONLY	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.

4 **Direct8x8InfFlag**

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
0	SUBBLOCK	allows subpartitioning to go below 8x8 block size (i.e. 4x4, 8x4 or 4x8)
1	BLOCK	allows processing only at 8x8 block size. MB Info is stored for 8x8 block size.

3 **Transform8x8Flag**

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
0	4x4	no 8x8 IDCT Transform, only 4x4 IDCT transform blocks are present
1	8x8	8x8 Transform is allowed

MFX_AVC_IMG_STATE

	2	**FrameMbOnlyFlag** 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** 0 / FALSE / not true ; effectively enables the possibility of MBAFF mode. 1 / TRUE / true, only frame MBs can occur in this sequence, hence disallows the MBAFF mode and field picture.
	1	**MbaffFlameFlag** 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** 0 / FALSE / not in MBAFF mode 1 / TRUE / in MBAFF mode
	0	**FieldPicFlag** 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 / Description** 0h / FRAME / a slice of a coded frame 1h / FIELD / a slice of a coded field
5 [ExistsIf]Encode Only	31	**Trellis Quantization Enabled (TQEnb)** Format: Enable The TQ improves output video quality of AVC CABAC encoder by selecting quantized values for each non-zero coefficient so as to minimize the total R-D cost.This flag is only valid AVC CABAC mode. Otherwise, this flag should be disabled. **Value / Name / Description** 0h / Disable / Use Normal 1h / Enable / Use Trellis quantization
	30:28	**Trellis Quantization Rounding (TQR)** This rounding scheme is only applied to the quantized coefficients ranging from 0 to 1 when TQEnb is set to 1 in AVC CABAC mode. One of the following values is added to quantized coefficients before truncating fractional part.

MFX_AVC_IMG_STATE

Value	Name	Description
000b		Add 1/8
001b		Add 2/8
010b		Add 3/8
011b		Add 4/8 (rounding 0.5)
100b		Add 5/8
101b		Add 6/8
110b	Default	Add 7/8 (Default rounding 0.875)

27 **Trellis Quantization Chroma Disable (TQChromaDisable)**

This signal is used to disable chroma TQ. To enable TQ for both luma and chroma, TQEnb=1, TQChromaDisable=0. To enable TQ only for luma, TQEnb=1, TQChromaDisable=1.

Value	Name	Description
0h		Enable Trellis Quantization chroma
1h	Default	Disable Trellis Quantization chroma

26:17 **Reserved**

Format:	MBZ

16 **NonFirstPassFlag**

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

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

15:13 **Reserved**

Format:	MBZ

12 **Reserved**

Format:	MBZ

11:10 **MinFrameWSizeUnits**

This field is the Minimum Frame Size Units

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

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

MB Rate Control conformance mask

MFX_AVC_IMG_STATE

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

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**

Format:	MBZ

7 **Intra/InterMbIpcmFlag - ForceIPCMControlMask**

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

Value	Name	Description
0h	Disable	Do not change intra or Inter macroblocks even
1h	Enable	Change intra or Inter 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**

Format:	MBZ

3 **FrameSzUnderFlag - FrameBitRateMinReportMask**

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

Value	Name	Description
0h	Disable	Do not update bit0 of MFC_IMAGE_STATUS control register.
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.

2 **FrameSzOverFlag - FrameBitRateMaxReportMask**

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

Value	Name	Description
0	Disable	Do not update bit0 of MFC_IMAGE_STATUS control register.
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.

MFX_AVC_IMG_STATE

	1	**InterMbMaxBitFlag - InterMBMaxSizeReportMask** This is a mask bit controlling if the condition of any inter MB in the frame exceeds InterMBMaxSize. **Value** / **Name** / **Description** 0 / Disable / Do not update bit0 of MFC_IMAGE_STATUS control register. 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.
	0	**IntraMbMaxBitFlag - IntraMBMaxSizeReportMask** This is a mask bit controlling if the condition of any intra MB in the frame exceeds IntraMBMaxSize. **Value** / **Name** / **Description** 0h / Disable / Do not update bit0 of MFC_IMAGE_STATUS control register. 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.
6 [ExistsIf]Encode Only	31:28	**Reserved**
	27:16	**InterMbMaxSz** Format: U12 This field, Inter MB Conformance Max size limit,indicates the allowed max bit count size for Inter MB
	15:12	**Reserved** Format: MBZ
	11:0	**IntraMbMaxSz** Exists If: //Intra Only Format: U12 This field, Intra MB Conformance Max size limit,indicates the allowed max bit count size for Intra MB All IPCM MBs should ignore this Max size limit.
7 [ExistsIf]Encode Only	31:17	**Reserved**
	16	**Reserved**
	15:1	**Reserved**
	0	**VSL Top MB Trans8x8flag** **Value** / **Name** / **Description** 0 / Disable **[Default]** / VSL will only fetch the current MB data.

MFX_AVC_IMG_STATE

		1 | Enable | When this bit is set VSL will make extra fetch to memory to fetch the MB data for top MB.
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]** 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).

MFX_AVC_IMG_STATE

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	31	**FrameBitrateMaxUnit** This field is the Frame Bitrate Maximum Limit Units.

MFX_AVC_IMG_STATE

Only

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

30 **FrameBitrateMaxUnitMode**

This field is the Frame Bitrate Maximum Limit Units.

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

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
0	Byte	FrameBitRateMax is in units of 32 Bytes when FrameBitrateMinUnitMode is 1 and in units of 128 Bytes if FrameBitrateMinUnitMode is 0
1	Kilo Byte	FrameBitRateMax is in units of 4KBytes Bytes when FrameBitrateMaxUnitMode is 1 and in units of 16KBytes if FrameBitrateMaxUnitMode is 0

14 **FrameBitrateMinUnitMode**

This field is the Frame Bitrate Minimum Limit Units.

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

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

MFX_AVC_IMG_STATE

		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	**Slice Stats Streamout Enable**
	30:16	**FrameBitRateMaxDelta** 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** 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	31:21	**Reserved** Format: MBZ
	20	**VMD Error Logic** **Value** / **Name** / **Description** 0 / Disable **[Default]** / 1 / Enable / Error Handling
	19	**Reserved** Format: MBZ
	18	**VAD Error Logic** **Value** / **Name** / **Description** 0 / Enable / Error reporting ON in case of premature Slice done

MFX_AVC_IMG_STATE

		[Default]	
		1	Disable	CABAC Engine will auto decode the bitstream in case of premature slice done.
	17	**Reserved**
	16	**Reserved**
	15:0	**Reserved** Format: MBZ
13	31:30	**Reserved** 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** 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: [0,31] — Name:
	15:14	**Reserved** 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: [0,31] — Name:

MFX_AVC_IMG_STATE

	7:0	**Initial QP Value** Format: S7 Range: [-26,25] 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.
14 [ExistsIf] Short Format only	31:24	**Log2_max_pic_order_cnt_lsb_minus4** Exists If: //Short Format Only It is a SPS syntax element, used to determine how many bits in the bitstream are used to represent pic_order_cnt_lsb syntax element in the slice header.Unsigned
	23:16	**Log2_max_frame_num_minus4** Exists If: //Short Format Only It is a SPS syntax element, used to determine how many bits in the bitstream are used to represent frame_num syntax element in the slice header.Unsigned.
	15	**deblocking_filter_control_present_flag** Exists If: //Short Format Only It is a PPS syntax element, indicates if more deblocking filter control syntax elements are present in the slice header.
	14:12	**num_slice_groups_minus1** Exists If: //Short Format Only BitField It is a PPS syntax element.Use for Slice Header parsing only, to read in slice_group_change_cycle, if any, but is not used by H/W, i.e. no slice group support.Desc
	11	**redundant_pic_cnt_present_flag** Exists If: //Short Format Only It is a PPS syntax element.Use for Slice Header parsing only, to read-in redundant_pic_cnt, if any, but is not used by H/W, i.e. no support for redundant slice processing.
	10:8	**slice_group_map_type** Exists If: //Short Format Only It is a PPS syntax element.Use for Slice Header parsing only, to read in slice_group_change_cycle, if any, but is not used by H/W, i.e. no slice group support.
	7:4	**Reserved** Format: MBZ

MFX_AVC_IMG_STATE

		IDR flag is decoded from NAL Header Byte
	3:2	**Pic_order_cnt_type** Exists If: //Short Format Only It is a SPS syntax element.Use for Slice Header parsing only.
	1	**Delta_pic_order_always_zero_flag** Exists If: //Short Format Only It is a SPS syntax element.Use for Slice Header parsing only.
	0	**Pic_order_present_flag** Exists If: //Short Format Only It is a PPS syntax element.Use for Slice Header parsing only.
15 [ExistsIf] Short Format only	31:16	**Curr Pic Frame Num** Exists If: //Short Format Only Format: U16 Derived from Slice Header syntax element
	15:0	**Slice Group Change Rate** Exists If: //Short Format Only Format: U16-1 It is a PPS syntax element Use for Slice Header parsing only, to read in slice_group_change_cycle, if any, but is not used by H/W, i.e. no slice group support.
16 [ExistsIf]: Short Format only	31	**Inter View Order Disable** Exists If: //Short Format Only It indicates how to append inter-view picture into initial sorted reference list. (due to ambiguity in the MVC Spec) Value / Name / Description: 0h / Default **[Default]** / View Order Ascending 1h / Disable / View ID Ascending
	30:22	**Reserved** Format: MBZ
	21:18	**Max View IDXL1** Exists If: //Short Format Only It is a PPS syntax element corresponding to Anchor/Non-Anchor Reference ListL1 It indicates the maximum number of inter-view picture for Reference List L1

MFX_AVC_IMG_STATE

	17:16	**Reserved** Format: MBZ
	15:12	**Max View IDXL0** Exists If: //Short Format Only Reference ListL0 It indicates the maximum number of inter-view picture for Reference List L0
	11:10	**Reserved** Format: MBZ
	9:0	**Current Frame View ID** Exists If: //Short Format Only It indicates the View ID of the current decoding frame
17	31:22	**Reserved** Format: MBZ
	21:16	**RhoDomain AverageMacroblockQP** Exists If: //[RhoDomain Rate Control] == 1 Format: U6
	15:9	**Reserved** Format: MBZ
	8	**Extended RhoDomain Statistics Enable** This parameter enables PAK to generate RhoDomain statistics from marcoblock QP and fractional QP computation. **Value** / **Name** / **Description** 0 / Disable / RhoDomain statistics generated from Frame Qp and no fractional QP computation. 1 / Enable / Enable MB QP based RhoDomain statistics and fractional QP computation.
	7:6	**Reserved** Format: MBZ
	5:3	**Fractional QP offset** Format: U3 Start position from the top of the Frame where increased Quantization parameter is added.
	2:0	**Fractional QP input** In a set of 8 rows, Qp is incremented by 1 for F_qp rows.
18	31:0	**Reserved** Format: MBZ

MFX_AVC_IMG_STATE		
19	31:0	**Threshold Size in Bytes** Format: U32 When a slice exceeds this value in bytes, hardware will end current slice as soon as possible and insert a new slice boundary. Note there is no guarantee that the actual slice size will meet this value, it is a hint to the HW to end the slice as soon as possible (which could be 2-5 macroblocks in the future from this detection point).
20	31:0	**Target Slice Size in Bytes** Format: U32

MFX_AVC_SLICE_STATE

MFX_AVC_SLICE_STATE

Source: VideoCS

Length Bias: 2

Description

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 Format: =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.

MFX_AVC_SLICE_STATE

		Value / **Name** 0000b — P Slice 0001b — B Slice 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** / **Name** 0-32
	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** / **Name** 0-32
	15:11	**Reserved** Format: MBZ
	10:8	**Log 2 Weight Denom Chroma** Format: U3 **Value** / **Name** 0-7
	7:3	**Reserved** Format: MBZ

MFX_AVC_SLICE_STATE

	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** 0 — Temporal; 1 — Spatial
	28:27	**Disable Deblocking Filter Indicator** **Value** 00b — **Name** — **Description** FilterInternalEdgesFlag is set equal to 1

MFX_AVC_SLICE_STATE

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**

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.

MFX_AVC_SLICE_STATE

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 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,

MFX_AVC_SLICE_STATE

		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

MFX_AVC_SLICE_STATE

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

18 **Reserved**

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

MFX_AVC_SLICE_STATE

		the current slice encoded bits.

16 SliceData Insertion Present in Bitstream

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**

MFX_AVC_SLICE_STATE

Encoder Only		Format: MBZ
	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: 0 - 512MB; Name:
8 Encoder Only	31:24	**Magnitude of QP Max Negative Modifier** Format: U8 This field specifies the lower limit of the QP modifier. Value: 0-51; Name:
	23:16	**Magnitude of QP Max Positive Modifier** Format: U8 This field specifies the upper limit of the QP modifier. Value: 0 - 15; Name:
	15:12	**Shrink Param - Shrink Resistance** Format: U4 This field specifies the additional points added each time decreased correction is invoked. Value: 0 - 15; Name:
	11:8	**Shrink Param - Shrink Init** Format: U4 This field specifies the initial points required to trip decreased control. Value: 0 - 15; Name:
	7:4	**Grow Param - Grow Resistance** Format: U4 This field specifies the additional points added each time increased correction is invoked. Value: 0 - 15; Name:

MFX_AVC_SLICE_STATE

	3:0	**Grow Param - Grow Init** Format: U4 This field specifies the initial points required to trip increased control. **Value** / **Name** 0 - 15
9 Encoder Only	31	**RoundInterEnable** Format: Enable When this bit is not set, RoundInter defaults to 2 to match SNB.
	30:28	**RoundInter** Format: U3 Rounding precision for Inter quantized coefficients **Value** / **Name** 000b: +1/16 **[Default]** 001b: +2/16 010b: +3/16 011b: +4/16 100b: +5/16 101b: +6/16 110b: +7/16 111b: +8/16
	27	**RoundIntraEnable** Format: Enable When this bit is not set, RoundIntra defaults to 4 to match SNB.
	26:24	**RoundIntra** Format: U3 Rounding precision for Intra quantized coefficients **Value** / **Name** 000b: +1/16 **[Default]** 001b: +2/16 010b: +3/16 011b: +4/16 100b: +5/16 101b: +6/16 110b: +7/16 111b: +8/16
	23:20	**Correct 6**

MFX_AVC_SLICE_STATE

		Format: U4 This field specifies the points used in the lowermost RC region when sum_act <= sum_min. Value: 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: 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: 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: 0 - 15
	7:4	**Correct 2** Format: U4 This field specifies the points used in the second RC region when sum_act > upper_midpt but <= sum_max. Value: 0 - 15
	3:0	**Correct 1** Format: U4 This field specifies the points used in the topmost RC region when sum_act > sum_max. Value: 0 - 15
10 Encoder Only	31:28	**ClampValues - CV7**
	27:24	**CV6**
	23:20	**CV5**
	19:16	**CV4**

MFX_AVC_SLICE_STATE

	15:12	**CV3**
	11:8	**CV2**
	7:4	**CV1**
	3:0	**CV0 - Clamp Value 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

MFX_AVC_SLICE_STATE

CV5	CV5	CV4	CV3	CV2	CV1	CV1	CV0

Value	Name
0 - 15	

MFX_BSP_BUF_BASE_ADDR_STATE

MFX_BSP_BUF_BASE_ADDR_STATE

Source: VideoCS

Length Bias: 2

This frame-level state command is used to specify all the buffer base addresses needed for the operation of the AVC Bit Stream Processing Units (for decoder, it is BSD Unit; for encoder, it is BSE Unit)

For both encoder and decoder, currently it is assumed that all codec standards can share the same BSP_BUF_BASE_STATE. The simplicity of this command is the result of moving all the direct MV related processing into the ENC Subsystem. Since all implicit weight calculations and directMV calculations are done in ENC and all picture buffer management are done in the Host, there is no need to provide POC (POC List - FieldOrderCntList, CurrPic POC - CurrFieldOrderCnt) information to PAK. For decoder, all the direct mode information are sent in a separate slice-level command (AVC_DIRECTMODE_STATE command).

In addition, in Encoder, the row stores for CABAC encoding and MB Parameters Construction (MPC) are combined into one single row store.

The row stores specified in this command do not combine with those specified in the MFC_PIPE_BUF_ADDR_STATE command for hardware simplification reason.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Pipeline 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: 4h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 8h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:6	**BSD/MPC Row Store Scratch Buffer Base Address - Read/Write**

MFX_BSP_BUF_BASE_ADDR_STATE

This field provides the base address of the scratch buffer used by BSD (decoder) and MPC (encoder) unit to store MB information of the previous row for coding each macroblock in the current row. It is a private buffer used by the BSD (decoder) and MPC (encoder) hardware only. Its content is not accessible by software. ThisRow Store buffer must be 64-byte cacheline aligned. Hardware uses the horizontal address of the current macroblock to address this Row Store.

For AVC BSD, 2 cacheline (CL) per MB when in MBAFF mode (row of MB pair); 1 CL per MB for non-MBAFF. So, to support 256 MBs per row (4K screen resolution), 2 * 256 * 64 bytes = 32,768 bytes are required. Cacheline alignment should be followed. For AVC MPC, 1 cachline for non-MBAFF, 2 cachelines for MBAFF per MB. For VC1, the BSD row store is 512-bit (one cacheline) per MB, times the number of MBs per picture MB row.

Programming Notes

This is one of the four RowStore Scratch Buffers which can programmed to use the internal Media Storage (total size 640 CacheLine). When Deblocking Filter Row Store Scratch Buffer Cache Select is programmed to "1", this will be stored inside MFX Media Internal Storage. Driver then needs to program this Base Address between 0 to 639, indicating starting cacheline address location for this buffer. Driver needs to make sure the whole buffer fits into Media Internal Storage.
(Notes: 1 cachelines per MB for non-mbaff; 2 cachelines per MB pair for mbaff, and the buffer needs to have enough space for 1 MB (pair) row).

	5:0	**Reserved** Format: MBZ
2	31:16	**Reserved** Format: MBZ Reserved for 64-bit address extension.
	15:0	**BSD/MPC Row Store Scratch Buffer Base Address - Read/Write [47:32]** **Description** This field is for the upper range of BSD/MPC Row Store Scratch Buffer Base Address. This field is used for 48-bit addressing.
3	31:15	**Reserved** Format: MBZ
	14:13	**BSD/MPC Row Store Scratch Buffer - Tiled Resource Mode** Format: U2 **For Media Surfaces:** This field specifies the tiled resource mode.

Value	Name	Description
0h	TRMODE_NONE	No tiled resource
1h	TRMODE_TILEYF	4KB tiled resources

MFX_BSP_BUF_BASE_ADDR_STATE

2h	TRMODE_TILEYS	64KB tiled resources
3h	Reserved	

12 — BSD/MPC Row Store Scratch Buffer Cache Select

This field controls if Intra Row Store is going to store inside Media Internal Storage or to LLC.

Value	Name	Description
0		Buffer going to LLC
1		Buffer going to Internal Media Storage

11 — Reserved

Format:	MBZ

10:9 — Reserved

Format:	MBZ

8:7 — BSD/MPC Row Store Scratch Buffer - Arbitration Priority Control

Format:	U2

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

6:1 — BSD/MPC Row Store Scratch Buffer - Index to Memory Object Control State (MOCS) Tables

Format:	U6

The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers.

The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.

0 — Reserved

4 — 31:6 — MPR Row Store Scratch Buffer Base Address - Read/Write (Decoder Only)

This field provides the base address of the scratch buffer used by decoder's MPR unit to store MB information of the previous row for decoding each macroblock in the current row. It is a private buffer used by the MPR hardware only. Its content is not accessible by software.

Programming Notes
The MPR Row Store buffer must be 64-byte cacheline aligned.Hardware uses the horizontal address of each macroblock to address the MPR Row Store. Except ILDB Control Data, all other operations does not cross slice boundary. This field is specified in frame-level.2 cacheline (CL) per MB when in MBAFF mode (row of MB pair); 1 CL per MB for non-MBAFF, So, to support 256 MBs per row (4K screen resolution), 2 * 256 * 64 bytes = 32,768 bytes are required. Cacheline alignment should be followed.This field is only valid for AVC decoder mode

MFX_BSP_BUF_BASE_ADDR_STATE

		This is one of the four RowStore Scratch Buffers which can programmed to use the internal Media Storage (total size 640 CacheLine). When Deblocking Filter Row Store Scratch Buffer Cache Select is programmed to "1", this will be cache inside MFX Media Internal Storage. Driver then needs to program this Base Address between 0 to 639, indicating starting cachelines address location for this buffer. Driver needs to make sure the whole buffer fits into Media Internal Storage *(Notes: 1 cachelines per MB for non-mbaff; 2 cachelines per MB pair for mbaff, and the buffer needs to have enough space for 1 MB (pair) row).*
	5:0	**Reserved** Format: MBZ
5	31:16	**Reserved** Format: MBZ Reserved for 64-bit address extension.
	15:0	**MPR Row Store Scratch Buffer Base Address - Read/Write [47:32]** This field is for the upper range of MPR Row Store Scratch Buffer Base Address. This field is used for 48-bit addressing.
6	31:15	**Reserved** Format: MBZ
	14:13	**MPR Row Store Scratch Buffer - Tiled Resource Mode** Format: U2 **For Media Surfaces:** This field specifies the tiled resource mode.

Value	Name	Description
0h	TRMODE_NONE	No tiled resource
1h	TRMODE_TILEYF	4KB tiled resources
2h	TRMODE_TILEYS	64KB tiled resources
3h	Reserved	

	12	**MPR Row Store Scratch Buffer Cache Select** This field controls if Intra Row Store is going to store inside Media Internal Storage or to LLC.

Value	Name	Description
0		Buffer going to LLC
1		Buffer going to Internal Media Storage

	11	**Reserved** Format: MBZ
	10:9	**Reserved** Format: MBZ
	8:7	**MPR Row Store Scratch Buffer - Arbitration Priority Control** Format: U2

MFX_BSP_BUF_BASE_ADDR_STATE

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

MPR Row Store Scratch Buffer - Index to Memory Object Control State (MOCS) Tables

Format:	U6

The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers.
The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.

Reserved

Bitplane Read Buffer Base Address
It must be cacheline aligned (i.e. 64 bytes address boundary), so lower bit 0 to 5 are used for controlling information.(In Cantiga, this field must be dword aligned.)Bitplane buffer is a linear buffer. In VC1 Long format, it is written by an application. In VC1 Short Format, it is written and read by H/W only.For VC1 intel Long Format : it is a read-only bufferFor VC1 DXVA2 Short Format : it is a write and a read bufferThis field is only valid for VC1 decoder mode.

Reserved

Format:	MBZ

Reserved

Format:	MBZ

Reserved for 64-bit address extension.

Bitplane Read Buffer Base Address - Read/Write [47:32]
This field is for the upper range of Bitplane Read Buffer Base Address.
This field is used for 48-bit addressing.

Reserved

Format:	MBZ

Bitplane Read Buffer - Tiled Resource Mode

Format:	U2

For Media Surfaces:
This field specifies the tiled resource mode.

Value	Name	Description
0h	TRMODE_NONE	No tiled resource
1h	TRMODE_TILEYF	4KB tiled resources
2h	TRMODE_TILEYS	64KB tiled resources
3h	Reserved	

MFX_BSP_BUF_BASE_ADDR_STATE

	12:11	**Reserved** Format: MBZ
	10:9	**Reserved** Format: MBZ
	8:7	**Bitplane Read Buffer - Arbitration Priority Control** Format: U2 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

	6:1	**Bitplane Read Buffer - Index to Memory Object Control State (MOCS) Tables** Format: U6 The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.
	0	**Reserved**

MI_ATOMIC

MI_ATOMIC

Source: BSpec

Length Bias: 2

Description

MI_ATOMIC is used to carry atomic operation on data in graphics memory. Atomic operations are supported on data granularity of 4B, 8B and 16B. The atomic operation leads to a read-modify-write operation on the data in graphics memory with the option of returning value. The data in graphics memory is modified by doing arithmetic and logical operation with the inline/indirect data provided with the MI_ATOMIC command. Inline/Indirect provided in the command can be one or two operands based on the atomic operation. Ex: Atomic-Compare operation needs two operands while Atomic-Add operation needs single operand and Atomic-increment requires no operand. Refer "Atomics" sub-section under "L3 Cache and URB" section for detailed atomic operations supported. Atomic operations can be enabled to return value by setting "Return Data Control" field in the command, return data is stored to CS_GPR registers.

CS_GPR4/5 registers are updated with memory Return Data based on the "Data Size". Each GPR register is qword in size and occupies two MMIO registers.

Note: Any references to CS_GPR registers in the command should be understood as the CS_GPR registers belonging to the corresponding engines *CS_GPR registers.

Engine Name	Corresponding GPR Registers
RCS	CS_GPR
BCS	BCS_GPR
VCS	VCS_GPR
VECS	VECS_GPR

Indirect Source Operands:

Operand1 is sourced from [CS_GPR1, CS_GPR0]

Operand2 is sourced from [CS_GPR3, CS_GPR2]

Read return Data is stored in [CS_GPR_5, CS_GPR4]

When "Data Size" is DWORD lower dword of CS_GPR4 (Qword) is updated with the dword data returned from memory. When "Data Size" is QWORD only CS_GPR4 (Qword) is updated with the qword data returned from memory. When the data size is OCTWORD CS_GPR4/5 are updated with the OCTWORD data returned from memory. CS_GPR4 is loaded with lower qword returned from memory and CS_GPR5 is loaded with upper qword returned from memory.

Programming Notes	Source
• When Inline Data mode is not set, Dwords 3..10 must not be included as part of the command. Dword Length field in the header must be programmed accordingly. • When Inline Data Mode is set, Dwords3..10 must be included based on the Data Size field of the header. Both Operand-1 and Operand-2 dwords must be programmed based on the Data Size field. Operand-2 must be programmed to 0x0 if the atomic operation doesn't require it. Dword Length field in the	

MI_ATOMIC

header must be programmed accordingly.	
Engines must be IDLE prior to command stream parsing MI_ATOMIC command. MI_FLUSH_DW prior to the MI_ATOMIC command would ensure the engine is IDLE prior to the atomic operation.	BlitterCS, VideoCS, VideoEnhancementCS

Workaround
Workaround: SW must ensure to program a PIPECONTROL command with CS Stall bit set prior to programming MI_ATOMIC command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 2Fh MI_ATOMIC Format: OpCode
	22	**Memory Type** This bit will be ignored and treated as if clear when executing from a non-privileged batch buffer. It is allowed for this bit to be clear when executing this command from a privileged (secure) batch buffer. This bit must be 1 if the **Per Process GTT Enable** bit is clear. Value 0h: Per Process Graphics Address Value 1h: Global Graphics Address — This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.
	21	**Reserved** Source: BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS Format: MBZ
	21	**Post-Sync Operation** Source: RenderCS Value 0h: No Post Sync Operation — Command is executed as usual. Value 1h: Post Sync Operation — MI_ATOMIC command is executed as a pipelined PIPE_CONTROL flush command with Atomics operation as post sync operation. Flush completion only guarantees the workload prior to this command is pushed till Windower unit and completion of any

MI_ATOMIC

				outstanding flushes issued prior to this command. When this bit set following ristiriciton apply to atomic operation: • Non-Compare atomic operations are supported on data granularity of 4B and 8B. DW3 is the lower dword of the operand and DW4 is the upper dword of the operand for the atomic operation. • Compare atomic operations are supported on data granularity of 4B. DW3 is Operand-0 and DW4 is Operand-1 for the atomic operation. • Atomic operations to GGTT/PPGTT memory surface are supported. • Only Inline data mode for atomic operand is supported, no support for indirect data mode. • No support for Return Data Control functionality. • No support for atomic operations on data granularity of 16B. • No support for compare atomic operations on data granularity of 8B.

Programming Notes
Any desired pipeline flush operation can be achieved by programming PIPE_CONTROL command prior to this command.
AWhen this bit is set Command Streamer sends a flush down the pipe and the atomic operation is saved as post sync operation. Command streamer goes on executing the following commands. Atomic operation saved as post sync operation is executed at some point later on completion of corresponding flush issued.
AWhen this bit is set atomic semaphore signal operation will be out of order with rest of the MI commands programmed in the ring buffer or batch buffer, it will be in order with respect to the post sync operations resulting due to PIPE_CONTROL command.

Workaround
PIPECONTROL command with “Command Streamer Stall Enable” must be programmed prior to programming MI_ATOMIC command with Post-Sync Operation set in GPGPU mode of operation (i.e when PIPELINE_SELECT command is set to GPGPU mode of operation).

20:19 **Data Size**

This field indicates the size of the operand in dword/qword/octword on which atomic operation will be performed. Data size must match with the Atomic Opcode. Operation Data size could be 4B, 8B or 16B

Value	Name	Description
0h	DWORD	Operand size used by Atomic Operation is DWORD.
1h	QWORD	Operand Size used by Atomic Operation is QWORD.

MI_ATOMIC

2h	OCTWORD	Operand Size used by Atomic Operation is OCTWORD.
3h	RESERVED	

18 Inline Data

This bit when set indicates the source operands are provided in line within the command. When reset the source operands are in CS_GPR registers.

Programming Notes
CS_GPR registers must be programmed with appropriate values before issuing MI_ATOMIC command with this field reset.

17 CS STALL

This bit when set command stream waits for completion of this command before executing the next command.

Programming Notes	Source
Render Command Streamer Only: CS will not guarantee atomic operation to be complete upon setting this bit along with Post Sync Operation set. When Post Sync Operation is set, this bit has no significance.	RenderCS

Workaround
Workaround: When CS STALL bit is set, Return Data Control must also be set in MI_ATOMIC command.

16 Return Data Control

Source:	RenderCS, BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS

Description
When Return Data Control is set the read return feature will be enabled during the atomic operation. Data is stored in CS_GPR5/4 registers unconditionally on completion of the atomic operation. On data return CS_GPR5/4 Registers are updated based on the "Data Size" field. When "Data Size" is DWORD lower dword of CS_GPR4 (Qword) is updated with the dword data returned from memory. When "Data Size" is QWORD only CS_GPR4 (Qword) is updated with the qword data returned from memory. When the data size is OCTWORD CS_GPR4/5 are updated with the OCTWORD data returned from memory. CS_GPR4 is loaded with lower qword returned from memory and CS_GPR5 is loaded with upper qword returned from memory.

Workaround
Workaround: When Return Data Control bit is set, CS STALL must also be set in MI_ATOMIC command.

15:8 ATOMIC OPCODE

This field selects the kind of atomic operation to be performed. Refer "Atomics" sub-section of "L3 Cache and URB" section of the B-spec for atomic opcode encoding and operation.

7:0 DWord Length

Format:	=n

MI_ATOMIC

Total Length - 2. Excludes DWord (0,1).

Value	Name	Exists If
1h	Inline Data 0 **[Default]**	([Inline Data]==0)
9h	Inline Data 1	([Inline Data]==1)

1	31:2	**Memory Address** Format: GraphicsAddress[31:2] This field contains the graphics memory address of the data on which atomic operation has to be performed. Atomic operation can be performed on data granularity of 4B, 8B or 16B and hence the Address has to be correspondingly aligned to 4B,8B or 16B respectively.
	1:0	**Reserved** Format: MBZ
2	31:16	**Reserved** Format: MBZ
	15:0	**Memory Address High** This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space.
3	31:0	**Operand1 Data Dword 0** Format: U32 Dword0 of Operand1 when Inline Data mode is set.
4	31:0	**Operand2 Data Dword 0** Format: U32 Dword0 of Operand2 when Inline Data mode is set.
5	31:0	**Operand1 Data Dword 1** Format: U32 Dword1 of Operand1 when Inline Data mode is set.
6	31:0	**Operand2 Data Dword 1** Format: U32 Dword1 of Operand2 when Inline Data mode is set.
7	31:0	**Operand1 Data Dword 2** Format: U32 Dword2 of Operand1 when Inline Data mode is set.
8	31:0	**Operand2 Data Dword 2** Format: U32 Dword2 of Operand2 when Inline Data mode is set.
9	31:0	**Operand1 Data Dword 3** Format: U32 Dword3 of Operand1 when Inline Data mode is set.

MI_ATOMIC		
10	31:0	**Operand2 Data Dword 3** Format: U32 Dword3 of Operand2 when Inline Data mode is set.

MI_SEMAPHORE_WAIT

MI_SEMAPHORE_WAIT

Source: CommandStreamer

Length Bias: 2

Description

This command supports memory based Semaphore WAIT. Memory based semaphores will be used for synchronization between the Producer and the Consumer contexts. Producer and Consumer Contexts could be running on different engines or on the same engine inside GT. Running on the same engine is only possible when execlists are enabled. Producer Context implements a Signal and Consumer context implements a Wait. Command Streamer on parsing this command fetches data from the Semaphore Address mentioned in this command and compares it with the inline Semaphore Data Dword.

- If comparison passes, the command streamer moves to the next command.
- When execlists are enabled, if comparison fails Command streamer switches out the context. Context switch can be inhibited by setting "Inhibit Synchronous Context Switch" in CTXT_SR_CTL register.
- In ring buffer mode of scheduling or Execlist with "Inhibit Synchronous context Switch", if comparison fails, Command Streamer evaluates the Compare Operation based on the Wait Mode until the compare operation is true or Wait is canceled by SW.
- Exec-List Scheduling: CS generates semaphore wait interrupt to the scheduler when MI_SEMAPHORE_WAIT command is un-successful and when "Inhibit Synchronous Context Switch" is set. Scheduler can use this interrupt to preempt the context waiting on semaphore wait.
- Ring Buffer Scheduling: CS generates semaphore wait interrupt to the scheduler when MI_SEMAPHORE_WAIT command is un-successful.

MI_SEMPHORE_SIGNAL and MI_SEMAPHORE_WAIT together replace the MI_SEMAPHORE_MBOX command.

MI_SEMAPHORE_WAIT command also supports register based Semaphore WAIT. Command Streamer on parsing this command fetches data from the MMIO offset mentioned in this command and compares it with the inline Semaphore Data Dword. This functionality is supported when “Register Poll” bit is set in the command header. In register poll mode of operation “Wait Mode” supported is always Poll mode and no Signal mode is supported.

- If comparison passes, the command streamer moves to the next command.
- Unlike in Memory based semaphore, there is no context switch or preemption supported in Register Poll mode of operation. Semaphore wait interrupt is not generated by default on wait un-successful in “Register Poll” mode.
- Register Poll mode of Semaphore Wait command operation is privileged and will not be supported from PPGTT batch buffers.
- HW will not trigger Render DOP CG on semaphore wait unsuccessful in register poll mode of operation.

DWord	Bit	Description	
0	31:29	**Command Type**	
		Default Value:	0h MI_COMMAND

MI_SEMAPHORE_WAIT

	Format:	OpCode

28:23 MI Command Opcode

Default Value:	1Ch MI_SEMAPHORE_WAIT
Format:	OpCode

22 Memory Type

This bit will be ignored and treated as if clear when executing from a non-privileged batch buffer. It is allowed for this bit to be clear when executing this command from a privileged (secure) batch buffer. This bit *must* be 1 if the **Per Process GTT Enable** bit is clear.

Value	Name	Description
0h	Per Process Graphics Address	
1h	Global Graphics Address	This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.

21:18 Reserved

Format:	MBZ

17 Reserved

Format:	MBZ

16 Register Poll Mode

This field control the seamphore wait behavior of polling from memory vs MMIO register.

Value	Name	Description
1h	Register Poll **[Default]**	In this mode HW periodically reads the semaphore data from MMIO register instead of memory for comparison until the condition is satisfied. Periodicity will be mentioned in a SEMA_WAIT_POLL register. When operating in register poll mode, DW2 “Semaphore Address” (bits 22:2) carries the register MMIO offset to be polled. In register poll mode “Memory Type” field of this command are ignored by HW.
0h	Memory Poll	In this mode HW will functional as in regular mode and checks for semaphore data in memory.

Programming Notes
In register poll mode of operation of MI_SEMAPHORE_WAIT command, context switch and preemption are not supported on wait un-successful. “Wait Mode” must be always set to “Polling Mode” when Register Poll Mode is enabled.

15 Wait Mode

This bit specifies the WAIT behavior when the semaphore comparison fails and before the context is switched out.

Value	Name	Description

MI_SEMAPHORE_WAIT

		1h	Polling Mode	In this mode HW periodically reads the semaphore data from memory for comparison until it is context switched out. Periodicity will be mentioned in a SEMA_WAIT_POLL register.
		0h	Signal Mode	In this mode HW will reacquire the semaphore data from memory on receiving SIGNAL with the same Context ID. In ring buffer mode of scheduling Context ID associated with SIGNAL is ignored and always treated as a match.

Programming Notes
Wait Mode must be always set to Polling Mode when Register Poll Mode is enabled.

14:12 Compare Operation

This field specifies the operation that will be executed to create the result that will either allow the context to continue or wait.

SAD = Semaphore Address Data
SDD = Semaphore Data Dword

Value	Name	Description
0h	SAD_GREATER_THAN_SDD	If Indirect fetched data is greater than inline data then continue.
1h	SAD_GREATER_THAN_OR_EQUAL_SDD	If Indirect fetched data is greater than or equal to inline data then continue.
2h	SAD_LESS_THAN_SDD	If Indirect fetched data is less than inline data then continue.
3h	SAD_LESS_THAN_OR_EQUAL_SDD	If Indirect fetched data is less than or equal to inline data then continue.
4h	SAD_EQUAL_SDD	If Indirect fetched data is equalto inline data then continue.
5h	SAD_NOT_EQUAL_SDD	If Indirect fetched data is not equal to inline data then continue.
6h	Reserved	
7h	Reserved	

11:8 Reserved

Format:	MBZ

7:0 DWord Length

Format:	=n Total Length - 2

Value	Name
2h	Excludes DWord (0,1) **[Default]**

1	31:0	**Semaphore Data Dword**

MI_SEMAPHORE_WAIT

		Format: U32 This Data dword is supplied by software to control execution of the command buffer. This value is used as part of the comparison to result in waiting or continuing in the command parser if enabled.
2..3	63:2	**Semaphore Address** Format: GraphicsAddress63-2 **Register Poll Mode:** In Register Poll mode of operation, Bits 22:2 (Bits 63:23 are reserved MBZ, HW enforced) specify the MMIO offset of the register for the semaphore. **Non Register Poll Mode:** This field is the Graphics Memory Address of the 32-bit value for the semaphore. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form.
	1:0	**Reserved** Format: MBZ

PIPE_CONTROL

PIPE_CONTROL

Source: RenderCS

Length Bias: 2

The PIPE_CONTROL command is used to effect the synchronization described above.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 2h PIPE_CONTROL Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 0h PIPE_CONTROL Format: OpCode
	15:10	**Reserved** Format: MBZ
	9	**Reserved**
	8	**Reserved**
	7:0	**DWord Length** Default Value: 4h DWORD_COUNT_n Format: =n Total Length - 2. Excludes DWord (0,1).
1	31	**Reserved** Format: MBZ
	30	**Reserved** Format: MBZ
	29	**Reserved** Format: MBZ
	28	**Reserved** Format: MBZ
	27	**Reserved**

PIPE_CONTROL

26 Flush LLC

Format:	Enable

If enabled, at the end of the current pipe-control the last level cache is cleared of all the cachelines which have been determined as being part of the Frame Buffer.

Programming Notes
SW must always program Post-Sync Operation to "Write Immediate Data" when Flush LLC is set.

25 Reserved

Format:	MBZ

24 Destination Address Type

Defines address space of Destination Address

Value	Name	Description
0h	PPGTT	Use PPGTT address space for DW write
1h	GGTT	Use GGTT address space for DW write

Programming Notes
Ignored if ""No Write" is selected in Operation.

23 LRI Post Sync Operation

Value	Name	Description
0h	No LRI Operation	No LRI operation occurs as a result of this instruction. The Post-Sync Operation field is valid and may be used to specify an operation.
1h	MMIO Write Immediate Data	Write the DWord contained in Immediate Data Low (DW3) to the MMIO offset specifed in the Address field.

Programming Notes
This bit caues a post sync operation with an LRI (Load Register Immediate) operation. If this bit is set then the Post-Sync Operation field must be cleared.

22 Reserved

21 Store Data Index

Format:	U1

Description
Ring Buffer Mode Scheduling: This field is valid only if the post-sync operation is not 0. If this bit is set, the store data address is actually an index into the global hardware status page. This bit only applies to the Global HW status page. If this field is 1, the Destination Address Type in this command must be set to 1 (GGTT). Execlist Mode Scheduling: This field is valid only if the post-sync operation is not 0. If this bit is set, the store data address is index into the global hardware status page when destination address type in the command is set to 1 (GGTT). The store data address is index into the per-process hardware status page when destination address type in the command is set to 0 (PPGTT).

PIPE_CONTROL

	20	**Command Streamer Stall Enable** Format: U1 If ENABLED, the sync operation will not occur until all previous flush operations pending a completion of those previous flushes will complete, including the flush produced from this command. This enables the command to act similar to the legacy MI_FLUSH command. **Programming Notes** A PIPE_CONTROL with Command Streamer Stall Enable set must be sent prior to enabling tessellation or a geometry shader.
	19	**Reserved**
	18	**TLB Invalidate** Format: U1 If ENABLED, all TLBs belonging to Render Engine will be invalidated once the flush operation is complete. Note that if the flush TLB invalidation mode is clear, a TLB invalidate will occur irrespective of this bit setting If ENABLED, PIPE_CONTROL command will flush the in flight data written out by render engine to Global Observation point on flush done. Also Requires stall bit ([20] of DW1) set. **Programming Notes** If ENABLED, all TLBs belonging to Render Engine will be invalidated once the flush operation is complete. Note that if the flush TLB invalidation mode is clear, a TLB invalidate will occur irrespective of this bit setting. Post Sync Operation or CS stall must be set to ensure a TLB invalidate occurs. Otherwise no cycle will occur to the TLB cache to invalidate.
	17	**Reserved** Format: MBZ
	16	**Generic Media State Clear** Format: Disable If set, all generic media state context information will be invalidated. Any state invalidated will not be saved as part of the render engine context image. The state only only become valid once it is parsed by the command streamer. **Workaround** Workaround: PIPECONTROL command with “Command Streamer Stall Enable” must be programmed prior to programming a PIPECONTROL command with "Media State Clear" set in GPGPU mode of operation (i.e when PIPELINE_SELECT command is set to GPGPU mode of operation).
	15:14	**Post Sync Operation** **Description** This field specifies an optional action to be taken upon completion of the synchronization operation.

PIPE_CONTROL

This field must be cleared if the LRI Post-Sync Operation bit is set.

Value	Name	Description
0h	No Write	No write occurs as a result of this instruction. This can be used to implement a "trap" operation, etc.
1h	Write Immediate Data	Write the QWord containing Immediate Data Low, High DWs to the Destination Address
2h	Write PS Depth Count	Write the 64-bit PS_DEPTH_COUNT register to the Destination Address
3h	Write Timestamp	Write the 64-bit TIMESTAMP register (i.e. "Reported Timestamp Count" 0x2358 for render pipe) to the Destination Address.

Programming Notes
If executed in non-secure batch buffer, the address given will be in a PPGTT address space. If in a secure ring or batch, address given will be in GGTT space

Workaround
Workaround: PIPECONTROL command with “Command Streamer Stall Enable” must be programmed prior to programming a PIPECONTROL command with Post Sync Op in GPGPU mode of operation (i.e when PIPELINE_SELECT command is set to GPGPU mode of operation).

13 **Depth Stall Enable**

Format:	Enable

This bit must be set when obtaining a "visible pixel" count to preclude the possible inclusion in the PS_DEPTH_COUNT value written to memory of some fraction of pixels from objects initiated after the PIPE_CONTROL command.

Value	Name	Description
0h	Disable	3D pipeline will not stall subsequent primitives at the Depth Test stage.
1h	Enable	3D pipeline will stall any subsequent primitives at the Depth Test stage until the Sync and Post-Sync operations complete.

Programming Notes
This bit must be DISABLED for operations other than writing PS_DEPTH_COUNT.
This bit will have no effect (besides preventing write cache flush) if set in a PIPE_CONTROL command issued to the Media pipe.

12 **Render Target Cache Flush Enable**

Format:	Enable

Setting this bit will force Render Cache to be flushed to memory prior to this synchronization point completing. This bit must be set for all write fence sync operations to assure that results from operations initiated prior to this command are visible in memory once software observes this synchronization.

PIPE_CONTROL

Value	Name	Description
0h	Disable Flush	Render Target Cache is NOT flushed.
1h	Enable Flush	Render Target Cache is flushed.

Programming Notes
This bit must be DISABLED for End-of-pipe (Read) fences, PS_DEPTH_COUNT or TIMESTAMP queries.
This bit must not be set when Depth Stall Enable bit is set in this packet.

11 **Instruction Cache Invalidate Enable**

Format:	Enable

Setting this bit is independent of any other bit in this packet. This bit controls the invalidation of the L1 and L2 at the top of the pipe i.e. at the parsing time.

10 **Texture Cache Invalidation Enable**

Format:	Enable

Setting this bit is independent of any other bit in this packet. This bit controls the invalidation of the texture caches at the top of the pipe i.e. at the parsing time.

Workaround
Workaround: "CS Stall" bit in PIPE_CONTROL command must be always set for GPGPU workloads when "Texture Cache Invalidation Enable" bit is set

9 **Indirect State Pointers Disable**

Format:	Enable

Description
At the completion of the post-sync operation associated with this pipe control packet, the indirect state pointers in the hardware are considered invalid; the indirect pointers are not saved in the context. If any new indirect state commands are executed in the command stream while the pipe control is pending, the new indirect state commands are preserved.
Using Invalidate State Pointer (ISP) only inhibits context restoring of Push Constant (3DSTATE_CONSTANT_*) commands. Push Constant commands are only considered as Indirect State Pointers. Once ISP is issued in a context, SW must initialize by programming push constant commands for all the shaders (at least to zero length) before attempting any rendering operation for the same context.

8 **Notify Enable**

Format:	Enable

If ENABLED, a Sync Completion Interrupt will be generated (if enabled by the MI Interrupt Control registers) once the sync operation is complete. See Interrupt Control Registers in Memory Interface Registers for details.

7 **Pipe Control Flush Enable**

PIPE_CONTROL

		Format: Enable Hardware on parsing PIPECONTROL command with Pipe Control Flush Enable set will wait for all the outstanding post sync operations corresponding to previously executed PIPECONTROL commands are complete before making forward progress.
	6	**Reserved**
	5	**DC Flush Enable** Format: Enable Setting this bit enables flushing of the L3$ portions that caches DC writes. **Programming Notes** DC Flush (L3 Flush) by default doesn't result in flushing/invalidating the IA Coherent lines from L3$, however this can be achieved by setting control bit "**Pipe line flush Coherent lines"** in "L3SQCREG4" register.
	4	**VF Cache Invalidation Enable** Format: Enable Setting this bit is independent of any other bit in this packet. This bit controls the invalidation of VF address based cache at the top of the pipe i.e. at the parsing time. **Programming Notes** If the VF Cache Invalidation Enable is set to a 1 in a PIPE_CONTROL, a separate Null PIPE_CONTROL, all bitfields sets to 0, with the VF Cache Invalidation Enable;set to 0 needs to be sent prior to the PIPE_CONTROL with VF Cache Invalidation Enable set to a 1. As the VF, VFR data caches do not support a full address tag, SW shall invalidate the VF, VFR cache before a draw call that uses modified VERTEX_BUFFER state. Failure to do so may result in erroneous hits in the cache. **Workaround** Workaround When VF Cache Invalidate is set "Post Sync Operation" must be enabled to "Write Immediate Data" or "Write PS Depth Count" or "Write Timestamp".
	3	**Constant Cache Invalidation Enable** Format: Enable Setting this bit is independent of any other bit in this packet. This bit controls the invalidation of the constant cache at the top of the pipe i.e. at the parsing time.
	2	**State Cache Invalidation Enable** Format: Enable Setting this bit is independent of any other bit in this packet. This bit controls the invalidation of the L1 and L2 state caches at the top of the pipe i.e. at the parsing time.
	1	**Stall At Pixel Scoreboard** Format: Enable Defines the behavior of PIPE_CONTROL command at the pixel scoreboard.

PIPE_CONTROL

Value	Name	Description
0h	Disable	Stall at the pixel scoreboard is disabled.
1h	Enable	Stall at the pixel scoreboard is enabled.

Programming Notes

This bit must be DISABLED for End-of-pipe (Read) fences, PS_DEPTH_COUNT or TIMESTAMP queries. This bit is ignored if Depth Stall Enable is set. Further the render cache is not flushed even if Write Cache Flush Enable bit is set.

	0	**Depth Cache Flush Enable** Format: Enable Setting this bit enables flushing (i.e. writing back the dirty lines to memory and invalidating the tags) of depth related caches. This bit applies to HiZ cache, Stencil cache and depth cache.

Value	Name	Description
0h	Flush Disabled	Depth relates caches (HiZ, Stencil and Depth) are NOT flushed.
1h	Flush Enabled	Depth relates caches (HiZ, Stencil and Depth) are flushed.

Programming Notes

Ideally depth caches need to be flushed only when depth is required to be coherent in memory for later use as a texture, source or honoring CPU lock. This bit must be DISABLED for End-of-pipe (Read) fences, PS_DEPTH_COUNT or TIMESTAMP queries.

2	31:2	**Address** Format: GraphicsAddress[31:2]U32 If **Post Sync Operation** is set to 1h ([DevIVB+]: **LRI Post-Sync Operation** must be clear): Bits 31:3 secify the QW address of where the Immediate Data following this DW in the packet to be stored. Bit 2 MBZ Ignored if "No Write" is the selected in Post-Sync Operation [DevIVB+]: If **LRI Post-Sync Operation** is set: Bits 22:2 (Bits 31:23 are reserved MBZ) specify the MMIO offset destination for the data in the **Immediate Data Low** (DW3) field. Only DW writes are valid.
	1:0	**Reserved** Format: MBZ
3	31:0	**Address High** Format: GraphicsAddress[63:32]U32 This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space. This field is valid only if the post-sync operation is not 0 and the LRI Post-Sync Operation is clear.
4..5	63:0	**Immediate Data** Format: U64 This field specifies the QWord value to be written to the targeted location. Only valid when Post-Sync Operation is 1h (Write Immediate Data) or LRI Post-Sync Operation is set. Ignored if Post-Sync Operation is "No write", "Write PS_DEPTH_COUNT" or "Write TIMESTAMP".

Split Send Message

sends - Split Send Message

Source: Eulsa

Length Bias: 4

Description

The sends instruction performs data communication between a thread and external function units, including shared functions (Sampler, Data Port Read, Data Port Write, URB, and Message Gateway) and some fixed functions (e.g. Thread Spawner, who also have an unique Shared Function ID). The sends instruction adds an entry to the EU's message request queue. The request message is stored in a block of contiguous GRF registers. The response message, if present, will be returned to a block of contiguous GRF registers. The return GRF writes may be in any order depending on the external function units. <src0> and <src1> are the lead GRF registers for the first and second block of the request respectively. <dest> is the lead GRF register for response. The message descriptor field <desc> contains the Message Length (the number of consecutive GRF registers corresponding to src0) and the Response Length (the number of consecutive GRF registers). It also contains the header present bit, and the function control signals. The extend message descriptor field <ex_desc> contains the target function ID, the Extended Message Length (the number of consecutive GRF registers corresponding to src1) and the extended function control signals. WrEn is forwarded to the target function in the message sideband.

The sends instruction is the only way to terminate a thread. When the EOT (End of Thread) bit of <ex_desc> is set, it indicates the end of thread to the EU, the Thread Dispatcher and, in most cases, the parent fixed function.

Message descriptor field <desc> can be a 32-bit immediate, imm32, or a 32-bit scalar register, <reg32a>. GEN restricts that the 32-bit scalar register <reg32a> must be the leading dword of the address register. It should be in the form of a0.0<0;1,0>:ud. When <desc> is a register operand, only the lower 31 bits of <reg32a> are used.

Extended Message descriptor field <ex_desc> can be a 32-bit immediate, imm32 only. The bits3:0 of the <ex_desc> specifies the SFID for the message.The EOT field always comes from bit127 of the instruction word, which is the bit5 of <ex_desc>. A thread must terminate with a sends instruction with EOT turned on. The bits9:6 of <ex_desc> specify the extended message length and bits31:16 specify the 16bit extended function control. Interpretation of the extended function control signals is subject to the target external function.</ex_desc></ex_desc></ex_desc></ex_desc>

<src0> is a 256-bit aligned GRF register. It serves as the leading GRF register of the request.
<src1> is a 256-bit aligned GRF register or a null register. It serves as the leading GRF register for the second block of the request when it is not a null register. It is required that the second block of GRFs does not overlap with the first block. If it is a null register the Extended Message Length must be 0. The sum of Message Length and Extended Message Length must not be greater than 15 on SKL.
The source dependency control, {NoSrcDepSet} is used to control the setting of source dependency for both the sources.
<dest> serves for two purposes: to provide the leading GRF register location for the response message if present, and to provide parameters to form the channel enable sideband signals.
<dest> signals whether there is a response to the message request. It can be either a null register, a direct-addressed GRF register or a register-indirect GRF register. Otherwise, hardware behavior is undefined.
If <dest> is null, there is no response to the request. Meanwhile, the Response Length field in <desc> must be 0. Certain types of message requests, such as memory write (store) through the Data Port, do not want

sends - Split Send Message

response data from the function unit. If so, the posted destination operand can be null.
If <dest> is a GRF register, the register number is forwarded to the shared function. In this case, the target function unit must send one or more response message phases back to the requesting thread. The number of response message phases must match the Response Length field in <desc>, which of course cannot be zero. For some cases, it could be an empty return message. An empty return message is defined as a single phase message with all channel enables turned off.
The destination type field is always valid and is used to generate the WrEn. This is true even if <dest> is a null register (this is an exception for null as for most cases these fields are ignored by hardware).
The address immediates for indirect sources and destination must be oword aligned.
The 16-bit channel enables of the message sideband are formed based on the WrEn. Interpretation of the channel enable sideband signals is subject to the target external function. In general for a 'sends' instruction with return messages, they are used as the destination dword write mask for the GRF registers starting at <dest>. For a message that has multiple return phases, the same set of channel enable signals applies to all the return phases.
NoDDClr and NoDDChk must not be used for send instruction.

Send a message stored in GRF locations starting at <src0> followed by <src1> to a shared function identified by <ex_desc> along with control from <desc> and <ex_desc> with a GRF writeback location at <dest>.

Format:
[(pred)] sends (exec_size) <dest> <src0> <src1> <ex_desc> <desc>

Restriction

Restriction : Software must obey the following rules in signaling the end of thread using the sends instruction: The posted destination operand must be null. No acknowledgement is allowed for the sends instruction that signifies the end of thread. This is to avoid deadlock as the EU is expecting to free up the terminated thread's resource. A thread must terminate with a sends instruction with message to a shared function on the output message bus; therefore, it cannot terminate with a sends instruction with message to the following shared functions: Sampler unit, NULL function For example, a thread may terminate with a URB write message or a render cache write message. A root thread originated from the media (generic) pipeline must terminate with a sends instruction with message to the Thread Spawner unit. A child thread should also terminate with a sends to TS. Please refer to the Media Chapter for more detailed description. The sends instruction can not update accumulator registers. Saturate is not supported for sends instruction. ThreadCtrl encodings Switch is not supported for sends instruction. The sends with EOT should use register space R112-R127 for <src>. This is to enable loading of a new thread into the same slot while the message with EOT for current thread is pending dispatch. Any instruction updating the ARF must use a {Switch} if the ARF is not used before EOT.</src>

Restriction : The source dependency control, {NoSrcDepSet}, must not be set for the send instruction preceding a send instruction with EOT.

Syntax

```
[(pred)] sends (exec_size) reg greg greg imm32 imm32 [(pred)] sends (exec_size) reg greg greg imm32 reg32a
```

Pseudocode

```
Evaluate(WrEn); <MsgChEnable> = WrEn; <SourceReg0> = <src0>.RegNum; <SourceReg1>
```

sends - Split Send Message

```
= <src1>.RegNum; MessageEnqueue(<MsgChEnable>, <ResponseReg>, <SourceReg0>,
<SourceReg1>,<ex_desc>, <dest>);
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

DWord	Bit	Description
0..3	127:96	**Message** Format: EU_INSTRUCTION_OPERAND_SEND_MSG
	95:80	**ExDesc[31:16]** Format: ExtMsgDescpt[31:16]
	79	**Source 0 Addressing Mode** Format: AddrMode
	78	**Reserved** Exists If: ([Source 0 Addressing Mode]=='Direct') Format: MBZ
	78	**Source 0 Address Immediate Sign [9]** Exists If: ([Source 0 Addressing Mode]=='Indirect') Format: S9[9]
	77	**SelReg32Desc**
	76:73	**Source 0 Address Subregister Number** Exists If: ([Source 0 Addressing Mode]=='Indirect')
	76:69	**Source 0 Register Number** Exists If: ([Source 0 Addressing Mode]=='Direct')
	72:68	**Source 0 Address Immediate [8:4]** Exists If: ([Source 0 Addressing Mode]=='Indirect') Format: S9[8:4]
	68	**Source 0 Subregister Number** Exists If: ([Source 0 Addressing Mode]=='Direct')
	67:64	**ExDesc[9:6]** Format: ExtMsgDescpt[9:6]
	63	**Destination Addressing Mode** Format: AddrMode
	62	**Destination Address Immediate Sign [9]** Exists If: ([Destination Addressing Mode]=='Indirect') Format: S9[9]

sends - Split Send Message

	62	**Reserved** Exists If: ([Destination Addressing Mode]=='Direct') Format: MBZ
	61	**Reserved** Format: MBZ
	60:57	**Destination Address Subregister Number** Exists If: ([Destination Addressing Mode]=='Indirect')
	60:53	**Destination Register Number** Exists If: ([Destination Addressing Mode]=='Direct')
	56:52	**Destination Address Immediate [8:4]** Exists If: ([Destination Addressing Mode]=='Indirect') Format: S9[8:4]
	52	**Destination Subregister Number [4]** Exists If: ([Destination Addressing Mode]=='Direct')
	51:44	**Source 1 Register Number**
	43:41	**Reserved** Format: MBZ
	40:37	**Destination Type**
	36	**Source 1 Register File** Format: RegFile[0]
	35	**Destination Register File** Format: RegFile[0]
	34	**MaskCtrl**
	33:32	**Flag Register Number/Subregister Number**
	31:28	**Controls B** Format: EU_INSTRUCTION_CONTROLS_B
	27:24	**Shared Function ID (SFID)** Format: SFID
	23:8	**Controls A** Format: EU_INSTRUCTION_CONTROLS_A
	7	**Reserved** Format: MBZ
	6:0	**Opcode** Format: EU_OPCODE