Intel® Open Source HD Graphics Programmers' Reference Manual (PRM)

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

For the 2014-2015 Intel Atom™ Processors, Celeron™ Processors and Pentium™ Processors based on the "Cherry Trail/Braswell" Platform (Cherryview/Braswell graphics)

June 2015, Revision 1.0

Creative Commons License

Notices and Disclaimers

Table of Contents

3DPRIMITIVE

3DPRIMITIVE

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The 3DPRIMITIVE command is used to submit 3D primitives to be processed by the 3D pipeline. Typically the processing results in rendering pixel data into the render targets, but this is not required.
The parameters passed in this command are forwarded to the Vertex Fetch function. The Vertex Fetch function will use this information to generate vertex data structures and store them in the URB. These vertices are then passed down the 3D pipeline.

Programming Notes

If the threads spawned by this command are required to observe memory writes performed by threads spawned from a previous command, software must precede this command with a command that performs a (preferably pipelined) memory flush (e.g., 3D_PIPECONTROL).

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: 3h 3DPRIMITIVE Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 0h 3DPRIMITIVE Format: OpCode
	15	**Reserved**
	14:13	**Reserved** Format: MBZ
	12	**Reserved**
	11	**Reserved**
	10	**Indirect Parameter Enable** Format: Enable If set, the values in DW 2-5 are ignored and replaced by the current values of the corresponding 3DPRIM_xxx MMIO registers: • 3DPRIM_VERTEX_COUNT (instead of DW2: VertexCountPerInstance)

3DPRIMITIVE

		• 3DPRIM_START_VERTEX (instead of DW3: StartVertexLocation) • 3DPRIM_INSTANCE_COUNT (instead of DW4: InstanceCount) • 3DPRIM_START_INSTANCE (instead of DW5: StartInstanceLocation) • 3DPRIM_BASE_VERTEX (instead of DW6: BaseVertexLocation) Indirect Parameter Enable and End Offset Enable shall not be ENABLED at the same time, or behavior is UNDEFINED.
	9	**UAV Coherency Required** Format: U1 SW will be required to set this bit if there is the possibility of sharing a UAV from a previous 3DPRIMITVE command. If set, this command may cause a flush due to UAV coherency requirements. If none of the shaders have UAV access enabled, then this bit is ignored.
	8	**Predicate Enable** Format: Enable If set, this command is executed (or not) depending on the current value of the MI Predicate internal state bit. This command is ignored only if PredicateEnable is set and the Predicate state bit is 0.
	7:0	**DWord Length** Default Value: 5h Excludes DWord (0,1) Project: CHV, BSW Format: =n Total Length - 2
1	31:10	**Reserved** Format: MBZ
	9	**End Offset Enable** Format: Enable If set, the Vertex Count Per Instance field is IGNORED, and the VB0ENDOFFSET register is used to indirectly specify the vertex count by defining the amount of valid data in VB0. The following restrictions apply: • VB0 must be enabled for use • VertexAccessType = SEQUENTIAL • Start Vertex Location = 0 • Start Instance Location = 0 • Base Vertex Location = 0 Vertices are output until EndOffset is reached or exceeded in VB0. If EndOffset is reached or exceeded within the data associated with a vertex, that vertex is considered incomplete and will not be output. Partial objects will be discarded (as is normally done).

3DPRIMITIVE

DWord	Bit	Description
		If clear, End Offset is ignored. Indirect Parameter Enable and End Offset Enable must not be ENABLED at the same time, or behavior is UNDEFINED.
	8	**Vertex Access Type** This field specifies how data held in vertex buffers marked as VERTEXDATA is accessed by Vertex Fetch. **Value** | **Name** | **Description** 0h | SEQUENTIAL | VERTEXDATA buffers are accessed sequentiallyRequiref if End Offset Enable is ENABLED. 1h | RANDOM | VERTEXDATA buffers are accessed randomly via an index obtained from the Index Buffer.
	7:6	**Reserved**
	5:0	**Primitive Topology Type** Format: 3D_Prim_Topo_Type [CHV, BSW] See table below for encoding, see 3D Overview for diagrams and general comments **Description** This field specifies the topology type of 3D primitive generated by this command. Note that a single primitive topology (list/strip/fan/etc.) can contain a number of basic objects (lines, triangles, etc.). This field is ignored. The topology type is specified via the 3DSTATE_VF_TOPOLOGY command.
2	31:0	**Vertex Count Per Instance** Format: U32 Count of vertices This field specifies how many vertices are to be generated for each instance of the primitive topology. If End Offset Enable is clear: Format = U32 count of vertices Range = [0, 2^32-1] (upper limit probably constrained by VB size) Ignored if End Offset Enable or Indirect Parameter Enable is ENABLED. **Programming Notes** • This per-instance value should specify a valid number of vertices for the primitive topology type. E.g., for 3DPRIM_TRILIST_ADJ, this field should specify a multiple of 6 vertices. However, in cases where too few or too many vertices are provided, the unused vertices will be silently discarded by the pipeline. • A 0 value is this field effectively makes the command a 'no-operation'.
3	31:0	**Start Vertex Location** Format: U32 structure index This field specifies the "starting vertex" for each instance. This allows skipping over part of the vertices in a buffer if, for example, a previous 3DPRIMITIVE command had already drawn the primitives associated with the earlier entries. For SEQUENTIAL access, this field specifies, for each instance, a starting structure index into the vertex buffers For RANDOM access, this field specifies, for each instance, a starting index into the Index Buffer.

<table>
<tr><th colspan="3">3DPRIMITIVE</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>

Access of any data outside of the valid extent of a vertex or index buffer will return the value 0 (i.e., appears as if the data stored at the invalid location was 0).
Must be set to 0 if End Offset Enable is ENABLED.
Ignored if Indirect Parameter Enable is ENABLED

</td></tr>
</table>
</td></tr>
<tr><td>4</td><td>31:0</td><td>
Instance Count
<table>
<tr><td>Format:</td><td>U32 Count of instances</td></tr>
</table>
This field specifies the number of instances by which the primitive topology is to be regenerated. A value of 0 indicates "no instances" (no-op operation). A value of 1 effectively specifies "non-instanced" operation, though vertex buffers will still be used to provide instance data, if so programmed. Ignored if Indirect Parameter Enable is ENABLED.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>[0, FFFFFFFFh]</td><td></td></tr>
</table>
</td></tr>
<tr><td>5</td><td>31:0</td><td>
Start Instance Location
<table>
<tr><td>Format:</td><td>U32 structure index</td></tr>
</table>
<table>
<tr><th>Description</th></tr>
<tr><td>This field specifies the "starting instance" for the command as an initial structure index into Vertex Buffers for vertex elements with InstancingEnable set.</td></tr>
<tr><td>Subsequent instances will access sequential instance data structures, as controlled by the Instance Data Step Rate.</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>

Access of any data outside of the valid extent of a vertex or index buffer will return the value 0 (i.e., appears as if the data stored at the invalid location was 0).
Must be set to 0 if End Offset Enable is ENABLED.
Ignored if Indirect Parameter Enable is ENABLED.

</td></tr>
</table>
</td></tr>
<tr><td>6</td><td>31:0</td><td>
Base Vertex Location
<table>
<tr><td>Format:</td><td>S31 index structure bias</td></tr>
</table>
This field specifies a signed bias to be added to values read from the index buffer. This allows the same index buffer values to access different vertex data for different commands.This field applies only to RANDOM access mode. This field is ignored for SEQUENTIAL access mode, where there Start Vertex Location can be used to specify different regions in the vertex buffers.
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>

Access of any data outside of the valid extent of a vertex or index buffer will return the value 0 (i.e., appears as if the data stored at the invalid location was 0).
Must be set to 0 if End Offset Enable is ENABLED.
Ignored if Indirect Parameter Enable is ENABLED.

</td></tr>
</table>
</td></tr>
</table>

3DSTATE_AA_LINE_PARAMETERS

3DSTATE_AA_LINE_PARAMETERS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The 3DSTATE_AA_LINE_PARAMS command is used to specify the slope and bias terms used in the improved alpha coverage computation (specifically for DX WHQL compliance). Note that in these devices the coverage values passed to PS threads are full U0.8 values.

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: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 0Ah 3DSTATE_AA_LINE_PARAMS Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**Dword Length** Default Value: 1h Excludes Dword (0,1) Project: All Format: =n Total Length - 2
1	31:24	**AA Point Coverage Bias** Project: CHV, BSW Format: U0.8 This field specifies the bias term to be used in the aa coverage computation for edges 0 and 3.
	23:16	**AA Coverage Bias** Project: All Format: U0.8

3DSTATE_AA_LINE_PARAMETERS		
		This field specifies the bias term to be used in the aa coverage computation for edges 0 and 3.
	15:8	**AA Point Coverage Slope** Project: CHV, BSW Format: U0.8 This field specifies the slope term to be used in the aa coverage computation for edges 0 and 3.If this field is zero, the Windower will revert to legacy aa line coverarge computation (though still output expanded U0.8 coverage values).
	7:0	**AA Coverage Slope** Project: All Format: U0.8 This field specifies the slope term to be used in the aa coverage computation for edges 0 and 3.If this field is zero, the Windower will revert to legacy aa line coverarge computation (though still output expanded U0.8 coverage values).
2	31:24	**AA Point Coverage EndCap Bias** Project: CHV, BSW Format: U0.8 This field specifies the bias term to be used in the aa coverage computation for edges 1 and 2.
	23:16	**AA Coverage EndCap Bias** Project: All Format: U0.8 This field specifies the bias term to be used in the aa coverage computation for edges 1 and 2.
	15:8	**AA Point Coverage EndCap Slope** Project: CHV, BSW Format: U0.8 This field specifies the slope term to be used in the aa coverage computation for edges 1 and 2.
	7:0	**AA Coverage EndCap Slope** Project: All Format: U0.8 This field specifies the slope term to be used in the aa coverage computation for edges 1 and 2.

3DSTATE_BINDING_TABLE_EDIT_DS

3DSTATE_BINDING_TABLE_EDIT_DS

Project: CHV, BSW
Source: RenderCS
Length Bias: 2

This command edits the binding table for DS.

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: 46h 3DSTATE_BINDING_TABLE_EDIT_DS Format: OpCode
	15:9	**Reserved** Format: MBZ
	8:0	**DWord Length** Format: =n Value / Name: 0h: DWORD_COUNT_n **[Default]** 0h - 100h: Range
1	31:16	**Binding Table Block Clear** Format: U16 Each bit in this field corresponds to a 16 entry block of the binding table. Bit 0 of this field corresponds to entries 0-15, bit 1 to 16-31, and so on. When a bit is set it clears the corresponding bind table entries to 0. (effectively disabling them). The clear is applied before the individual binding table entries contained in this message are applied. When this bit is clear then the corresponding 16 entry block is not cleared.
	15:2	**Reserved** Format: MBZ

3DSTATE_BINDING_TABLE_EDIT_DS

	1:0	**Binding Table Edit Target** Specifies which core should respond to this **3DSTATE_BINDING_TABLE_EDIT_DS** command:
2..n	31:0	**Entry [n]** Format: BINDING_TABLE_EDIT_ENTRY [CHV, BSW]

Value	Name	Description
11b	All Cores	All cores should respond to this command
10b	Core 1	Only Core1 should respond to this command
01b	Core 0	Only Core0 should respond to this command
00b	Reserved	Reserved

3DSTATE_BINDING_TABLE_EDIT_GS

3DSTATE_BINDING_TABLE_EDIT_GS		
Project:	CHV, BSW	
Source:	RenderCS	
Length Bias:	2	
This command edits the binding table for GS.		

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: 44h 3DSTATE_BINDING_TABLE_EDIT_GS Format: OpCode
	15:9	**Reserved** Format: MBZ
	8:0	**DWord Length** Format: =n Value / Name: 0h — DWORD_COUNT_n **[Default]** 0h - 100h — Range
1	31:16	**Binding Table Block Clear** Format: U16 Each bit in this field corresponds to a 16 entry block of the binding table. Bit 0 of this field corresponds to entries 0-15, bit 1 to 16-31, and so on. When a bit is set it clears the corresponding bind table entries to 0. (effectively disabling them). The clear is applied before the individual binding table entries contained in this message are applied. When this bit is clear then the corresponding 16 entry block is not cleared.
	15:2	**Reserved** Format: MBZ

3DSTATE_BINDING_TABLE_EDIT_GS		
	1:0	**Binding Table Edit Target** Specifies which core should respond to this **3DSTATE_BINDING_TABLE_EDIT_GS** command: <table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>11b</td><td>All Cores</td><td>All cores should respond to this command</td></tr><tr><td>10b</td><td>Core 1</td><td>Only Core1 should respond to this command</td></tr><tr><td>01b</td><td>Core 0</td><td>Only Core0 should respond to this command</td></tr><tr><td>00b</td><td>Reserved</td><td>Reserved</td></tr></table>
2..n	31:0	**Entry [n]** <table><tr><td>Format:</td><td>BINDING_TABLE_EDIT_ENTRY [CHV, BSW]</td></tr></table>

3DSTATE_BINDING_TABLE_EDIT_HS

3DSTATE_BINDING_TABLE_EDIT_HS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command edits the binding table for HS.

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: 45h 3DSTATE_BINDING_TABLE_EDIT_HS Format: OpCode
	15:9	**Reserved** Format: MBZ
	8:0	**DWord Length** Format: =n Value / Name: 0h — DWORD_COUNT_n **[Default]** 0h - 100h — Range
1	31:16	**Binding Table Block Clear** Format: U16 Each bit in this field corresponds to a 16 entry block of the binding table. Bit 0 of this field corresponds to entries 0-15, bit 1 to 16-31, and so on. When a bit is set it clears the corresponding bind table entries to 0. (effectively disabling them). The clear is applied before the individual binding table entries contained in this message are applied. When this bit is clear then the corresponding 16 entry block is not cleared.
	15:2	**Reserved** Format: MBZ

3DSTATE_BINDING_TABLE_EDIT_HS		
	1:0	**Binding Table Edit Target** Specifies which core should respond to this **3DSTATE_BINDING_TABLE_EDIT_HS** command: 11b — All Cores — All cores should respond to this command 10b — Core 1 — Only Core1 should respond to this command 01b — Core 0 — Only Core0 should respond to this command 00b — Reserved — Reserved
2..n	31:0	**Entry [n]** Format: BINDING_TABLE_EDIT_ENTRY [CHV, BSW]

Binding Table Edit Target values:

Value	Name	Description
11b	All Cores	All cores should respond to this command
10b	Core 1	Only Core1 should respond to this command
01b	Core 0	Only Core0 should respond to this command
00b	Reserved	Reserved

3DSTATE_BINDING_TABLE_EDIT_PS

3DSTATE_BINDING_TABLE_EDIT_PS

Project: CHV, BSW
Source: RenderCS
Length Bias: 2

This command edits the binding table for PS.

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: 47h 3DSTATE_BINDING_TABLE_EDIT_PS Format: OpCode
	15:9	**Reserved** Format: MBZ
	8:0	**DWord Length** Format: =n Value / Name: 0h — DWORD_COUNT_n **[Default]** 0h - 100h — Range
1	31:16	**Binding Table Block Clear** Format: U16 Each bit in this field corresponds to a 16 entry block of the binding table. Bit 0 of this field corresponds to entries 0-15, bit 1 to 16-31, and so on. When a bit is set it clears the corresponding bind table entries to 0. (effectively disabling them). The clear is applied before the individual binding table entries contained in this message are applied. When this bit is clear then the corresponding 16 entry block is not cleared.
	15:2	**Reserved** Format: MBZ

3DSTATE_BINDING_TABLE_EDIT_PS		
	1:0	**Binding Table Edit Target** Specifies which core should respond to this **3DSTATE_BINDING_TABLE_EDIT_PS** command:

Value	Name	Description
11b	All Cores	All cores should respond to this command
10b	Core 1	Only Core1 should respond to this command
01b	Core 0	Only Core0 should respond to this command
00b	Reserved	Reserved

2..n	31:0	**Entry [n]** Format: BINDING_TABLE_EDIT_ENTRY [CHV, BSW]

3DSTATE_BINDING_TABLE_EDIT_VS

3DSTATE_BINDING_TABLE_EDIT_VS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command edits the binding table for VS.
The 3DSTATE_BINDING_TABLE_EDIT_VS is a variable length command.

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: 43h 3DSTATE_BINDING_TABLE_EDIT_VS Format: OpCode
	15:9	**Reserved** Format: MBZ
	8:0	**DWord Length** Format: =n Value / Name: 0h: DWORD_COUNT_n **[Default]** 0h - 100h: Range
1	31:16	**Binding Table Block Clear** Format: U16 Each bit in this field corresponds to a 16 entry block of the binding table. Bit 0 of this field corresponds to entries 0-15, bit 1 to 16-31, and so on. When a bit is set it clears the corresponding bind table entries to 0. (affectively disabling them). The clear is applied before the individual binding table entries contained in this message are applied. When this bit is clear then the corresponding 16 entry block is not cleared.
	15:2	**Reserved** Format: MBZ

3DSTATE_BINDING_TABLE_EDIT_VS		
	1:0	**Binding Table Edit Target** Specifies which core should respond to this **3DSTATE_BINDING_TABLE_EDIT_VS** command: <table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>11b</td><td>All Cores</td><td>All cores should respond to this command</td></tr><tr><td>10b</td><td>Core 1</td><td>Only Core1 should respond to this command</td></tr><tr><td>01b</td><td>Core 0</td><td>Only Core0 should respond to this command</td></tr><tr><td>00b</td><td>Reserved</td><td>Reserved</td></tr></table>
2..n	31:0	**Entry [n]** Format: BINDING_TABLE_EDIT_ENTRY [CHV, BSW]

3DSTATE_BINDING_TABLE_POINTERS_DS

3DSTATE_BINDING_TABLE_POINTERS_DS		
Project:	CHV, BSW	
Source:	RenderCS	
Length Bias:	2	
The 3DSTATE_BINDING_TABLE_POINTERS_DS command is used to define the location of fixed functions' BINDING_TABLE_STATE. Only some of the fixed functions utilize binding tables.		
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: 28h 3DSTATE_BINDING_TABLE_POINTERS_DS Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Project: All Format: =n
1	31:16	**Reserved** Project: All Format: MBZ
	15:5	**Pointer to DS Binding Table** Project: CHV, BSW Format: SurfaceStateOffset[15:5]BINDING_TABLE_STATE*256 When HW binding table is disabled Format: SurfaceStateOffset[16:6]BINDING_TABLE_STATE*256 When HW-generated binding table is enabled

3DSTATE_BINDING_TABLE_POINTERS_DS		
		Specifies an aligned address offset of the function's BINDING_TABLE_STATE. The offset's base and alignment differ depending on whether HW Binding Table is enabled: If HW Binding Table is disabled, the offset is relative to **Surface State Base Address** and the alignment is **32B**. If HW Binding Table is enabled the offset is relative to the **Binding Table Pool Base Address** and the alignment is **64B**.
	4:0	**Reserved** Project: All Format: MBZ

3DSTATE_BINDING_TABLE_POINTERS_GS

3DSTATE_BINDING_TABLE_POINTERS_GS		
Project:	CHV, BSW	
Source:	RenderCS	
Length Bias:	2	
The 3DSTATE_BINDING_TABLE_POINTERS_GS command is used to define the location of fixed functions' BINDING_TABLE_STATE. Only some of the fixed functions utilize binding tables.		
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: 29h 3DSTATE_BINDING_TABLE_POINTERS_GS Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Project: All Format: =n
1	31:16	**Reserved** Project: All Format: MBZ
	15:5	**Pointer to GS Binding Table** Project: CHV, BSW Format: SurfaceStateOffset[15:5]BINDING_TABLE_STATE*256 When HW binding table is disabled Format: SurfaceStateOffset[16:6]BINDING_TABLE_STATE*256 When HW-generated binding table is enabled

3DSTATE_BINDING_TABLE_POINTERS_GS		
		Specifies an aligned address offset of the function's BINDING_TABLE_STATE. The offset's base and alignment differ depending on whether HW Binding Table is enabled: If HW Binding Table is disabled, the offset is relative to **Surface State Base Address** and the alignment is **32B**. If HW Binding Table is enabled the offset is relative to the **Binding Table Pool Base Address** and the alignment is **64B**.
	4:0	**Reserved** Project: All Format: MBZ

3DSTATE_BINDING_TABLE_POINTERS_HS

3DSTATE_BINDING_TABLE_POINTERS_HS		
Project:	CHV, BSW	
Source:	RenderCS	
Length Bias:	2	
The 3DSTATE_BINDING_TABLE_POINTERS_HS command is used to define the location of fixed functions' BINDING_TABLE_STATE. Only some of the fixed functions utilize binding tables.		
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: 27h 3DSTATE_BINDING_TABLE_POINTERS_HS Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Project: All Format: =n
1	31:16	**Reserved** Project: All Format: MBZ
	15:5	**Pointer to HS Binding Table** Project: CHV, BSW Format: SurfaceStateOffset[15:5]BINDING_TABLE_STATE*256 When HW binding table is disabled Format: SurfaceStateOffset[16:6]BINDING_TABLE_STATE*256 When HW-generated binding table is enabled

3DSTATE_BINDING_TABLE_POINTERS_HS

		Specifies an aligned address offset of the function's BINDING_TABLE_STATE. The offset's base and alignment differ depending on whether HW Binding Table is enabled: If HW Binding Table is disabled, the offset is relative to **Surface State Base Address** and the alignment is **32B**. If HW Binding Table is enabled the offset is relative to the **Binding Table Pool Base Address** and the alignment is **64B**.
	4:0	**Reserved** Project: All Format: MBZ

3DSTATE_BINDING_TABLE_POINTERS_PS

3DSTATE_BINDING_TABLE_POINTERS_PS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The 3DSTATE_BINDING_TABLE_POINTERS_PS command is used to define the location of fixed functions' BINDING_TABLE_STATE. Only some of the fixed functions utilize binding tables.

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: 2Ah 3DSTATE_BINDING_TABLE_POINTERS_PS Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Project: All Format: =n
1	31:16	**Reserved** Project: All Format: MBZ
	15:5	**Pointer to PS Binding Table** Project: CHV, BSW Format: SurfaceStateOffset[15:5]BINDING_TABLE_STATE*256 When HW binding table is disabled Format: SurfaceStateOffset[16:6]BINDING_TABLE_STATE*256 When HW-generated binding table is enabled

3DSTATE_BINDING_TABLE_POINTERS_PS		
		Specifies an aligned address offset of the function's BINDING_TABLE_STATE. The offset's base and alignment differ depending on whether HW Binding Table is enabled: If HW Binding Table is disabled, the offset is relative to **Surface State Base Address** and the alignment is **32B**. If HW Binding Table is enabled the offset is relative to the **Binding Table Pool Base Address** and the alignment is **64B**.
	4:0	**Reserved** Project: All Format: MBZ

3DSTATE_BINDING_TABLE_POINTERS_VS

3DSTATE_BINDING_TABLE_POINTERS_VS		
Project:	CHV, BSW	
Source:	RenderCS	
Length Bias:	2	
The 3DSTATE_BINDING_TABLE_POINTERS_VS command is used to define the location of fixed functions' BINDING_TABLE_STATE. Only some of the fixed functions utilize binding tables.		
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: 26h 3DSTATE_BINDING_TABLE_POINTERS_VS Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Project: All Format: =n
1	31:16	**Reserved** Project: All Format: MBZ
	15:5	**Pointer to VS Binding Table** Project: CHV, BSW Format: SurfaceStateOffset[15:5]BINDING_TABLE_STATE*256 When HW binding table is disabled Format: SurfaceStateOffset[16:6]BINDING_TABLE_STATE*256 When HW-generated binding table is enabled

3DSTATE_BINDING_TABLE_POINTERS_VS

		Specifies an aligned address offset of the function's BINDING_TABLE_STATE. The offset's base and alignment differ depending on whether HW Binding Table is enabled: If HW Binding Table is disabled, the offset is relative to **Surface State Base Address** and the alignment is **32B**. If HW Binding Table is enabled the offset is relative to the **Binding Table Pool Base Address** and the alignment is **64B**.
	4:0	**Reserved** Project: All Format: MBZ

3DSTATE_BINDING_TABLE_POOL_ALLOC

3DSTATE_BINDING_TABLE_POOL_ALLOC

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command sets up the binding table pool for HW generated binding tables.

Programming Notes

When RS is enabled due to a MI_RS_CONTROL or MI_BATCH_BUFFER_START with RS enable bit set, driver must reprogam the 3DSTATE_BINDING_TABLE_POOL_ALLOC to ensure the resource streamer and render engine are in sync with the programming with the command. Otherwise there could be cases where RS sees that the Binding Table Pool is disabled while the render pipeline sees the binding table is enabled in the case the 3DSTATE_BINDING_TABLE_POOL_ALLOC was enabled while RS was off.

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: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 19h 3DSTATE_BINDING_TABLE_POOL_ALLOC Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Project: All Format: =n Value: 2h; Name: DWORD_COUNT_n **[Default]**; Project: CHV, BSW
1..2 **Project:** CHV,	63:12	**Binding Table Pool Base Address** Project: CHV, BSW

3DSTATE_BINDING_TABLE_POOL_ALLOC

BSW		Format: GraphicsAddress[63:12]BindingTablePool This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space.
	11	**Binding Table Pool Enable** Project: CHV, BSW Format: U1 When this bit is set it enables HW generation of binding tables. When this bit is cleared it disables HW generation of binding tables.
	10	**Reserved** Project: CHV, BSW
	9:7	**Reserved** Project: CHV, BSW
	6:0	**Surface Object Control State** Project: CHV, BSW Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this surface. **Programming Notes** Bit 2 is not programmable and is always zero.
3 **Project:** CHV, BSW	31:12	**Binding Table Pool Buffer Size** Project: CHV, BSW Format: U20 This field specifies the size of the buffer in 4K pages. Any access which straddle or go past the end of the buffer will return 0. Value | Name | Description [0,1048575] | | 0 | No Valid Data | There is no valid data in the buffer **Restriction** Programming size of zero is illegal in the case that the pool is enabled.
	11	**Reserved** Project: CHV, BSW Format: MBZ
	10:0	**Reserved** Project: CHV, BSW Format: MBZ

3DSTATE_BLEND_STATE_POINTERS

3DSTATE_BLEND_STATE_POINTERS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The 3DSTATE_BLEND_STATE_POINTERS command is used to set up the pointers to the color calculator state.

Programming Notes
When the BLEND_STATE pointer changes but not the CC_STATE pointer, driver needs to force a CC_STATE pointer change to improve blend performance in pixel backend.

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: 24h 3DSTATE_BLEND_STATE_POINTERS Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Project: All Format: =n
1	31:6	**Blend State Pointer** Project: All Format: DynamicStateOffset[31:6]BLEND_STATE*8 Specifies the 64-byte aligned offset of the BLEND_STATE. This offset is relative to the **Dynamic State Base Address**.
	5:1	**Reserved**

3DSTATE_BLEND_STATE_POINTERS		
		Project: All Format: MBZ
	0	**Blend State Pointer Valid** Project: CHV, BSW Format: Enable This bit, if set, indicates that the BLEND_STATE pointer has changed and new state needs to be fetched.

3DSTATE_CC_STATE_POINTERS

3DSTATE_CC_STATE_POINTERS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The 3DSTATE_CC_STATE_POINTERS command is used to set up the pointers to the color calculator state.

Programming Notes

When the CC_STATE pointer changes but not the BLEND_STATE pointer, driver needs to force a BLEND_STATE pointer change in order to improve blend performance in the pixel backend.

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:	0Eh 3DSTATE_CC_STATE_POINTERS
		Format:	OpCode
	15:8	**Reserved**	
		Project:	All
		Format:	MBZ
	7:0	**DWord Length**	
		Default Value:	0h DWORD_COUNT_n
		Project:	All
		Format:	=n
1	31:6	**Color Calc State Pointer**	
		Project:	All
		Format:	DynamicStateOffset[31:6]COLOR_CALC_STATE
		Specifies the 64-byte aligned offset of the COLOR_CALC_STATE. This offset is relative to the **Dynamic State Base Address**.	
	5:1	**Reserved**	

3DSTATE_CC_STATE_POINTERS		
		Project: All Format: MBZ
	0	**Color Calc State Pointer Valid** Project: CHV, BSW Format: Enable If set, the hardware will fetch the CC state. This bit is context saved and restored so the CC state is considered undefined once this bit is cleared due to the possiblity of the CC state changing between context switches.

3DSTATE_CHROMA_KEY

3DSTATE_CHROMA_KEY

Project: CHV, BSW
Source: RenderCS
Length Bias: 2

The 3DSTATE_CHROMA_KEY instruction is used to program texture color/chroma-key key values. A table containing four set of values is supported. The ChromaKey Index sampler state variable is used to select which table entry is associated with the map. Texture chromakey functions are enabled and controlled via use of the ChromaKey Enable texture sampler state variable.Texture Color Key (keying on a paletted texture index) is not supported.

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: 1h 3DSTATE_NONPIPELINED Format: Opcode
	23:16	**3D Command Sub Opcode** Default Value: 04h 3DSTATE_CHROMA_KEY Format: Opcode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 2h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:30	**ChromaKey Table Index** Project: All Format: U2 index Selects which entry in the ChromaKey table is to be loaded Value: [0,3] — Name:

3DSTATE_CHROMA_KEY

DWord	Bit	Description
	29:0	**Reserved** Project: All Format: MBZ
2	31:0	**ChromaKey Low Value** This field specifies the "low" (minimum) value of the chroma key range. Texel samples are considered "matching the key" if each component of the texel falls within the (inclusive) chroma range.See ChromaKey High Value for further format, programming info.
3	31:0	**ChromaKey High Value** This field specifies the "high" (maximum) value of the chroma key range. Texel samples are considered "matching the key" if each component of the texel falls within the (inclusive) chroma range.

Programming Notes

ChromaKey values are specified using 8-bit channels. When using surface formats with less than 8 bits per channel, the device will expand channels by replicating the required number of MSBs into the LSBs of each channel. Software must account for this conversion when it programs Chromakey Low/High Values (e.g., by performing the same replication).

For channels that do not exist in the actual surface (e.g., Alpha channel for non-ARGB maps), software must explicitly program full range high/low values (High=FFh, Low=0h for formats using unsigned chroma key values, High=7Fh, Low=FFh for formats using sign magnitude chroma key values) in order to effectively remove the comparison of that field from the ChromaKey function.

For channels in SNORM format in the surface format, the value in the high/low value for that channel is interpreted in sign magnitude format. Negative zero value is not supported (use positive zero instead). For channels with mixed UNORM/SNORM formats (i.e. R5G5_SNORM_B6_UNORM), the ChromaKey is programmed as if all channels are SNORM.

YUV ChromaKey will use an interpolated chrominance value from the map for comparison to the chroma key values for those texels without chrominance due to downsampling. The chrominance value used is the average of values to the left and right of the texel in question.

It is UNDEFINED to program any component of the ChromaKey High Value to be less than the corresponding component of ChromaKey Low Value.

Format = interpreted according to associated texel format "class":

Only the surface formats listed as supported for chroma key in the surface formats table can be used with this feature. Use of any other surface format with chroma key enabled is UNDEFINED.

Surface Format	31:24	23:15	16:8	7:0
ARGB and BC (DXT) formats	A	R	G	B
YCrCb formats	A	Cr	Y	Cb

3DSTATE_CLEAR_PARAMS

3DSTATE_CLEAR_PARAMS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

Description	Project
This command defines the depth clear value delivered as a pipelined state command. However, the state change pipelining isn't completely transparent (see restriction below).	
HW will internally manage the draining pipe and flushing of the caches when this command is issued. The PIPE_CONTROL restrictions are removed.	CHV, BSW

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: 04h 3DSTATE_CLEAR_PARAMS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**Dword Length** Default Value: 1h Excludes Dword (0,1) Format: =n Total Length - 2
1	31:0	**Depth Clear Value** Project: CHV, BSW Format: IEEE_Float This field defines the clear value that will be applied to the depth buffer if the Depth Buffer Clear field is enabled. It is valid only if Depth Buffer Clear Value Valid is set. **Programming Notes** The clear value must be between the min and max depth values (inclusive) defined in the CC_VIEWPORT. If the depth buffer format is D32_FLOAT, then values must be limited to the

3DSTATE_CLEAR_PARAMS		
		range of +0.0f and 1.0f inclusive; values outside this range are reserved.
2	31:1	**Reserved** Format: MBZ
	0	**Depth Clear Value Valid** Format: Boolean This field enables the **Depth Clear Value**. If clear, the depth clear value is obtained from interpolated depth of an arbitrary pixel of the primitive rendered with **Depth Buffer Clear** set in WM_STATE or 3DSTATE_WM. If set, the depth clear value is obtained from the **Depth Clear Value** field of this command.

3DSTATE_CLIP

3DSTATE_CLIP

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

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: 12h 3DSTATE_CLIP Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 02h Excludes DWord (0,1) Project: All Format: =n Total Length - 2
1	31:21	**Reserved** Project: All Format: MBZ
	20	**Force User Clip Distance Cull Test Enable Bitmask** Project: All Format: Enable This field provides a work around override for the computation of SOL_INT::Render_Enable Value | Name | Description 0h | Normal | Clip_INT::User Clip Distance Cull Test Enable Bitmask normally

3DSTATE_CLIP

1h	Force	Forces Clip_INT::User Clip Distance Cull Test Enable Bitmask to use the value in 3DSTATE_CLIP:: User Clip Distance Cull Test Enable Bitmask

19 — Vertex Sub Pixel Precision Select

Project:	All
Format:	U1

Selects the number of fractional bits maintained in the vertex data

Value	Name	Description	Project
0h	8 Bit	8 sub pixel precision bits maintained	All
1h	4 Bit	4 sub pixel precision bits maintained	All

18 — Early Cull Enable

Project:	All
Format:	Enable

This field is used to enable/disable the EarlyCull function.

17 — Force User Clip Distance Clip Test Enable Bitmask

Project:	All
Format:	Enable

This field provides a work around override for the computation of SOL_INT::Render_Enable.

Value	Name	Description	Project
0b	Normal	Clip_INT:: User Clip Distance Clip Test Enable Bitmask normally	All
1b	Force	Forces Clip_INT:: User Clip Distance Clip Test Enable Bitmask to use the value in 3DSTATE_CLIP::User Clip Distance Clip Test Enable Bitmask	All

16 — Force Clip Mode

Format:	Enable

This field provides a work around override for the computation of SOL_INT::Render_Enable.

Value	Name	Description	Project
0b	Normal	Clip_INT::Clip Mode is computed normally.	All
1b	Force	Forces Clip_INT::Clip Mode to use the value in 3DSTATE_CLIP::User Clip Mode.	All

15:11 — Reserved

Project:	All
Format:	MBZ

10 — Clipper Statistics Enable

Project:	All
Format:	Enable

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

Value	Name	Description	Project

3DSTATE_CLIP

		<table><tr><td>0h</td><td>Disable</td><td>CL_INVOCATIONS_COUNT cannot increment</td><td>All</td></tr><tr><td>1h</td><td>Enable</td><td>CL_INVOCATIONS_COUNT can increment</td><td>All</td></tr></table>
	9:8	**Reserved** <table><tr><td>Project:</td><td></td></tr><tr><td>Format:</td><td>MBZ</td></tr></table>
	7:0	**User Clip Distance Cull Test Enable Bitmask** <table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>Enable[8]</td></tr></table>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.
2	31	**Clip Enable** <table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>Enable</td></tr></table>Specifies whether the Clip function is enabled or disabled (pass-through).
	30	**API Mode** <table><tr><td>Project:</td><td>All</td></tr></table>Controls the definition of the NEAR clipping plane <table><tr><th>Value</th><th>Name</th><th>Description</th><th>Project</th></tr><tr><td>0h</td><td>OGL</td><td>NEAR VP boundary == 0.0 (NDC)</td><td>All</td></tr></table>
	29	**Reserved** <table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table>
	28	**Viewport XY Clip Test Enable** <table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>Enable</td></tr></table>This field is used to control whether the Viewport X, Y extents are considered in VertexClipTest. See Tristrip Clipping subsection.
	27	**Reserved** <table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table>
	26	**Guardband Clip Test Enable** <table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>Enable</td></tr></table>This field is used to control whether the Guardband X, Y extents are considered in VertexClipTest

3DSTATE_CLIP

		for non-point objects. If the Guardband ClipTest is DISABLED but the Viewport XY ClipTest is ENABLED, ClipDetermination operates as if the Guardband were coincident with the Viewport. If both the Guardband and Viewport XY ClipTest are DISABLED, all vertices are considered "visible" with respect to the XY directions.
	25:24	**Reserved** Project: All Format: MBZ
	23:16	**User Clip Distance Clip Test Enable Bitmask** Project: All 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.
	15:13	**Clip Mode** Project: All This field specifies a general mode of the CLIP unit, when the CLIP unit is ENABLED.

Value	Name	Description	Project
0h	NORMAL	TrivialAccept objects are passed down the pipeline, MustClip objects Clipped in the Fixed Function Clipper HW, TrivialReject and BAD objects are discarded	All
1h	Reserved		All
2h	Reserved		All
3h	REJECT_ALL	All objects are discarded	All
4h	ACCEPT_ALL	All objects (except BAD objects) are trivially accepted. This effectively disables the clip-test/clip-determination function. Note that the CLIP unit will still filter out adacency information, which may be required since the SF unit does not accept primitives with adjacency.	All
5h-7h	Reserved		All

	12:10	**Reserved** Project: All Format: MBZ
	9	**Perspective Divide Disable** Project: All Format: Disable This field disables the Perspective Divide function performed on homogeneous position read from the URB. This feature can be used by software to submit pre-transformed "screen-space" geometry for rasterization. This likely requires the W component of positions to contain "rhw"

3DSTATE_CLIP

		(aka 1/w) in order to support perspective-correct interpolation of vertex attributes. Likewise, the X, Y, Z components will likely be required to be X/W, Y/W, Z/W. Note that the device does not support clipping when perspective divide is disabled. Software must specify CLIPMODE_ACCEPT_ALL whenever it disables perspective divide. This implies that software must ensure that object positions are completely contained within the "guardband" screen-space limits imposed by the SF unit (e.g., by clipping in CPU SW before submitting the objects).
	8	**Non-Perspective Barycentric Enable** Project: All Format: Enable This field enables computation of non-perspective barycentric parameters in the clipper, which are sent to SF unit in the must clip case. This field must be enabled if any non-perspective barycentric parameters are enabled in the Windower.
	7:6	**Reserved** Project: All Format: MBZ
	5:4	**Triangle Strip/List Provoking Vertex Select** Project: All Format: U2 enumerated type This field selects which vertex of a triangle (in a triangle strip or list primitive) is considered the "provoking vertex".

Value	Name	Project
0h	0	All
1h	1	All
2h	2	All
3h	Reserved	All

	3:2	**Line Strip/List Provoking Vertex Select** Project: All Format: U2 enumerated type This field selects which vertex of a line (in a line strip or list primitive) is considered the "provoking vertex".

Value	Name	Project
0h	0	All

3DSTATE_CLIP

DWord	Bit	Description
		1h | 1 | All 2h | Reserved | All 3h | Reserved | All
	1:0	**Triangle Fan Provoking Vertex Select** Project: All Format: U2 enumerated type This field selects which vertex of a triangle (in a triangle fan primitive) is considered the "provoking vertex". **Value | Name | Project** 0h | 0 | All 1h | 1 | All 2h | 2 | All 3h | Reserved | All
3	31:28	**Reserved** Project: All Format: MBZ
	27:17	**Minimum Point Width** Project: All Format: U8.3 pixels This value is used to clamp read-back PointWidth values.
	16:6	**Maximum Point Width** Project: All Format: U8.3 pixels This value is used to clamp read-back PointWidth values.
	5	**Force Zero RTA Index Enable** Project: All Format: Enable If set, the Clip unit will ignore the read-back RTAIndex and operate as if the value 0 was read-back. If clear, the read-back value is used.
	4	**Reserved** Project: All Format: MBZ

3DSTATE_CLIP		
	3:0	**Maximum VP Index** Project: All Format: U4-1 index value (# of viewports) This field specifies the maximum valid VPIndex value, corresponding to the number of active viewports. If the source of the VPIndex exceeds this maximum value, a VPIndex value of 0 is passed down the pipeline. Note that this clamping does not affect a VPIndex value stored in the URB.

3DSTATE_CONSTANT_DS

3DSTATE_CONSTANT_DS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command sets pointers to the push constants for the DS unit. The constant data pointed to by this command is loaded into the DS unit's push constant buffer (PCB).

Programming Notes	Project
[CHV, BSW] A 3DSTATE_GATHER_DS command must be dispatched along with any 3DSTATE_CONSTANT_DS command when Gather Pool is enabled within a batch buffer.	CHV, BSW

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: 1Ah 3DSTATE_CONSTANT_DS Format: OpCode
	15	**Reserved** Project: CHV, BSW Format: MBZ
	14:8	**Constant Buffer Object Control State** Project: CHV, BSW Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for all constant buffers defined in this command. **Programming Notes** / **Project**: Constant Buffer Object Control State must be always programmed to zero. — CHV, BSW
	7:0	**DWord Length** Project: All

3DSTATE_CONSTANT_DS

		Format: =n Total Length - 2 **Value** | **Name** | **Project** 9h | Excludes DWord (0,1) **[Default]** | CHV, BSW
1..10 **Project:** CHV, BSW	319:0	**Constant Body** Project: CHV, BSW Format: 3DSTATE_CONSTANT(Body) Following table is the shared portion of the 3DSTATE_CONSTANT command for VS, HS, DS, and GS

3DSTATE_CONSTANT_GS

3DSTATE_CONSTANT_GS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command sets pointers to the push constants for the GS unit. The constant data pointed to by this command will be loaded into the GS unit's push constant buffer (PCB).

Programming Notes	Project
[CHV, BSW]: A 3DSTATE_GATHER_GS command must be dispatched along with any 3DSTATE_CONSTANT_GS command when the Gather Pool is enabled within a batch buffer.	CHV, BSW

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: 16h 3DSTATE_CONSTANT_GS Format: OpCode
	15	**Reserved** Project: All Format: MBZ
	14:8	**Constant Buffer Object Control State** Project: CHV, BSW Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for all constant buffers defined in this command. Programming Notes: Constant Buffer Object Control State must be always programmed to zero. (Project: CHV, BSW)
	7:0	**DWord Length** Project: All

3DSTATE_CONSTANT_GS

		Format: =n Total Length - 2 **Value** | **Name** | **Project** 9h | Excludes DWord (0,1) **[Default]** | CHV, BSW
1..10 **Project:** CHV, BSW	319:0	**Constant Body** Project: CHV, BSW Format: 3DSTATE_CONSTANT(Body) Following table is the shared portion of the 3DSTATE_CONSTANT command for VS, HS, DS, and GS

3DSTATE_CONSTANT_HS

3DSTATE_CONSTANT_HS	
Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command sets pointers to the push constants for the HS unit. The constant data pointed to by this command is loaded into the HS unit's push constant buffer (PCB).

Programming Notes	Project
A 3DSTATE_GATHER_HS command must be dispatched along with any 3DSTATE_CONSTANT_HS command when Gather Pool is enabled within a batch buffer.	CHV, BSW

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: 19h 3DSTATE_CONSTANT_HS Format: OpCode
	15	**Reserved** Project: CHV, BSW Format: MBZ
	14:8	**Constant Buffer Object Control State** Project: CHV, BSW Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for all constant buffers defined in this command. Programming Notes: Constant Buffer Object Control State must be always programmed to zero. (Project: CHV, BSW)
	7:0	**DWord Length** Project: All

3DSTATE_CONSTANT_HS		
		Format: =n Total Length - 2 **Value** | **Name** | **Project** 9h | Excludes DWord (0,1) **[Default]** | CHV, BSW
1..10 **Project:** CHV, BSW	319:0	**Constant Body** Project: CHV, BSW Format: 3DSTATE_CONSTANT(Body) Following table is the shared portion of the 3DSTATE_CONSTANT command for VS, HS, DS, and GS

3DSTATE_CONSTANT_PS

3DSTATE_CONSTANT_PS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command sets pointers to the push constants for the PS unit. The constant data pointed to by this command is loaded into the PS unit's push constant buffer (PCB).

Programming Notes	Project
[CHV, BSW]: A 3DSTATE_GATHER_PS command must be dispatched along with any 3DSTATE_CONSTANT_PS command when the Gather Pool is enabled within a batch buffer.	CHV, BSW

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: 17h 3DSTATE_CONSTANT_PS Format: OpCode
	15	**Reserved** Project: CHV, BSW Format: MBZ
	14:8	**Constant Buffer Object Control State** Project: CHV, BSW Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for all constant buffers defined in this command. Programming Notes: Constant Buffer Object Control State must be always programmed to zero. (Project: CHV, BSW)
	7:0	**Dword Length** Project: All

3DSTATE_CONSTANT_PS		
		Format: =n Total Length - 2 Value: 9h — Name: Excludes DWord (0,1) **[Default]** — Project: CHV, BSW
1..10 **Project:** CHV, BSW	319:0	**Constant Body** Project: CHV, BSW Format: 3DSTATE_CONSTANT(Body) Following table is the shared portion of the 3DSTATE_CONSTANT command for VS, HS, DS, and GS

3DSTATE_CONSTANT_VS

3DSTATE_CONSTANT_VS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command sets pointers to the push constants for VS unit. The constant data pointed to by this command is loaded into the VS unit's push constant buffer (PCB).

Programming Notes	Project
[CHV, BSW] A 3DSTATE_GATHER_VS command must be dispatched along with any 3DSTATE_CONSTANT_VS command when Gather Pool is enabled within a batch buffer.	CHV, BSW

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: 15h 3DSTATE_CONSTANT_VS Format: OpCode
	15	**Reserved** Project: CHV, BSW Format: MBZ
	14:8	**Constant Buffer Object Control State** Project: CHV, BSW Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for all constant buffers defined in this command. **Programming Notes** / **Project**: Constant Buffer Object Control State must be always programmed to zero. — CHV, BSW
	7:0	**DWord Length** Project: All

<table>
<tr><th colspan="3">3DSTATE_CONSTANT_VS</th></tr>
<tr><td></td><td></td><td>
<table><tr><td>Format:</td><td>=n Total Length - 2</td></tr></table>
<table><tr><th>Value</th><th>Name</th><th>Project</th></tr>
<tr><td>9h</td><td>Excludes DWord (0,1) [Default]</td><td>CHV, BSW</td></tr></table>
</td></tr>
<tr><td>1..10
Project: CHV, BSW</td><td>319:0</td><td>Constant Body
<table><tr><td>Project:</td><td>CHV, BSW</td></tr>
<tr><td>Format:</td><td>3DSTATE_CONSTANT(Body)</td></tr></table>
Following table is the shared portion of the 3DSTATE_CONSTANT command for VS, HS, DS, and GS</td></tr>
</table>

3DSTATE_DEPTH_BUFFER

3DSTATE_DEPTH_BUFFER

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The depth buffer surface state is delivered as a pipelined state packet. However, the state change pipelining isn't completely transparent (see restriction below).

WM HW will internally manage the draining pipe and flushing of the caches when this commands is issued. The PIPE_CONTROL restrictions are removed.

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: 5h 3DSTATE_DEPTH_BUFFER Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 6h Excludes Dword (0,1) Format: =n Excludes DWord(0,1)
1	31:29	**Surface Type**

Value	Name	Description
0h	SURFTYPE_1D	Defines a 1-dimensional map or array of maps
1h	SURFTYPE_2D	Defines a 2-dimensional map or array of maps
2h	SURFTYPE_3D	Defines a 3-dimensional (volumetric) map
3h	SURFTYPE_CUBE	Defines a cube map
4h-6h	Reserved	

3DSTATE_DEPTH_BUFFER

7h	SURFTYPE_NULL	Defines a null surface

Programming Notes

The Surface Type of the depth buffer must be the same as the Surface Type of the render target(s) (defined in SURFACE_STATE), unless either the depth buffer or render targets are SURFTYPE_NULL.

28 — Depth Write Enable

Format:	Enable

This field enables depth writes to the depth buffer surface. Both this field and the Depth Buffer Write Enable field in DEPTH_STENCIL_STATE must be enabled in order for depth writes to occur.

27 — Stencil Write Enable

Format:	Enable

This field enables stencil writes to the depth buffer or stencil buffer surface, depending on where stencil is located. Both this field and the Stencil Buffer Write Enable field in DEPTH_STENCIL_STATE must be enabled in order for stencil writes to occur.

26:23 — Reserved

Format:	MBZ

22 — Hierarchical Depth Buffer Enable

Format:	Enable

If enabled, indicates that a hierarchical depth buffer is defined.

Programming Notes

If this field is enabled, the Software Tiled Rendering Mode must be NORMAL. This field must be disabled if Early Depth Test Enable is disabled OR if depth buffer surface type is NULL.

21 — Reserved

Format:	MBZ

20:18 — Surface Format

Specifies the format of the depth buffer. See Stencil Test Enable field in DEPTH_STENCIL_STATE field for restrictions on the use of some of these formats.

Value	Name
0h	Reserved
1h	D32_FLOAT
2h	Reserved
3h	D24_UNORM_X8_UINT
4h	Reserved
5h	D16_UNORM
6h-7h	Reserved

3DSTATE_DEPTH_BUFFER

DWord	Bit	Description
	17:0	**Surface Pitch** Format: U18-1 Pitch in (Bytes-1)

Value	Name	Description
[7Fh,3FFFFh]		corresponding to [128B, 256KB] also restricted to a multiple of 128B

Programming Notes
The pitch specified must be a multiple of the tile pitch, in the range [128B, 128KB].

DWord	Bit	Description
2..3 **Project:** CHV, BSW	63:0	**Surface Base Address** Project: CHV, BSW Format: GraphicsAddress[63:0]DepthBuffer This field specifies address of the buffer in mapped Graphics Memory. Graphics Address [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] = [47].

Programming Notes
The Depth Buffer can only be mapped to Main Memory (uncached).
If the buffer is linear, the surface must be 64-byte aligned.

DWord	Bit	Description
4	31:18	**Height** Format: U14-1 This field specifies the height of the surface. If the surface is MIP-mapped, this field contains the height of the base MIP level.

Value	Name	Description	Exists If
[0,0]	Legal Range	Must be zero	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_1D')
[0,16383]	Legal Range	Height of surface - 1 (y/v dimension)	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_2D')
[0,2047]	Legal Range	Height of surface - 1 (y/v dimension)	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_3D')
[0,16383]	Legal Range	y/v dimension	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_CUBE')

Programming Notes
The Height of the depth buffer must be the same as the Height of the render target(s) (defined in SURFACE_STATE), unless Surface Type is SURFTYPE_1D or SURFTYPE_2D with Depth = 0 (non-array) and LOD = 0 (non-mip mapped).

DWord	Bit	Description
	17:4	**Width** Format: U14-1 This field specifies the width of the surface. If the surface is MIP-mapped, this field specifies the width of the base MIP level. The width is specified in units of pixels.

3DSTATE_DEPTH_BUFFER

Value	Name	Description	Exists If
[0,16383]	Legal Range	Width of surface - 1 (x/u dimension)	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_1D')
[0,16383]	Legal Range	Width of surface - 1 (x/u dimension)	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_2D')
[0,2047]	Legal Range	Width of surface - 1 (x/u dimension)	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_3D')
[0,16383]	Legal Range	Width of surface - 1 (x/u dimension)	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_CUBE')

Programming Notes
The Width specified by this field must be less than or equal to the surface pitch (specified in bytes via the Surface Pitch field). For cube maps, Width must be set equal to Height. The Width of the depth buffer must be the same as the Width of the render target(s) (defined in SURFACE_STATE), unless Surface Type is SURFTYPE_1D or SURFTYPE_2D with Depth = 0 (non-array) and LOD = 0 (non-mip mapped).

3:0 **LOD**

Format:	U4 for LOD units

Value	Name
[0,14]	

Programming Notes
The LOD of the depth buffer must be the same as the LOD of the render target(s) (defined in SURFACE_STATE)

5 31:21 **Depth**

Format:	U11-1

This field specifies the total number of levels for a volume texture or the number of array elements allowed to be accessed starting at the Minimum Array Element for arrayed surfaces. If the volume texture is MIP-mapped, this field specifies the depth of the base MIP level.

Value	Name	Description	Exists If
[0,2047]	Legal Range	Number of array elements - 1	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_1D')
[0,2047]	Legal Range	Number of array elements - 1	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_2D')
[0,2047]	Legal Range	Depth of surface - 1 (r/z dimension)	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_3D')
[0,0]	Legal Range	Must be zero	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_CUBE')

3DSTATE_DEPTH_BUFFER

		Programming Notes The Depth of the depth buffer must be the same as the Depth of the render target(s) (defined in SURFACE_STATE).
	20:10	**Minimum Array Element** Format: U11 **For 1D and 2D Surfaces:** This field indicates the minimum array element that can be accessed as part of this surface. The delivered array index is added to this field before being used to address the surface. **For 3D Surfaces** This field indicates the minimum 'R' coordinate on the LOD currently being rendered to. This field is added to the delivered array index before it is used to address the surface. **For Other Surfaces** This field is ignored Value | Name | Exists If [0,2047] | SURFTYPE_1D/2D | (Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_1D'|Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_2D') [0,2047] | SURFTYPE_3D | (Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_3D')
	9:7	**Reserved** Format: MBZ
	6:0	**Depth Buffer Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for the depth buffer.
6	31:26	**Reserved** Project: CHV, BSW Format: MBZ
	25:0	**Reserved** Format: MBZ
7 **For 1D and 2D Surfaces:** This field must be set to the same value as the Depth	31:21	**Render Target View Extent** Format: U11-1 Value | Name | Description | Exists If [0,2047] | Legal Range | Number of array elements- 1 | (Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_1D') [0,2047] | Legal Range | Number of array elements- 1 | (Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_2D') [0,2047] | Legal Range | To indication extent of [1,2048] | (Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_3D')

<table>
<tr><th colspan="3">3DSTATE_DEPTH_BUFFER</th></tr>
<tr>
<td rowspan="3">field.
For 3D Surfaces: This field indicates the extent of the accessible 'R' coordinates minus 1 on the LOD currently being rendered to.
For Other Surfaces This field is ignored.</td>
<td></td>
<td>
<table>
<tr><td>[0,0]</td><td>Legal Range</td><td>Must be zero</td><td>(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_CUBE')</td></tr>
</table>
</td>
</tr>
<tr><td>20:15</td><td>Reserved</td></tr>
<tr>
<td>14:0</td>
<td>Surface QPitch
<table>
<tr><td>Format:</td><td>QPitch[16:2]</td></tr>
</table>
<table>
<tr><th>Description</th></tr>
<tr><td>This field specifies the distance in rows between array slices. It is used only in the following cases:

Surface Array is enabled OR
Number of Mulitsamples is not NUMSAMPLES_1 and Multisampled Surface Storage Format set to MSFMT_MSS OR
Surface Type is SURFTYPE_CUBE
</td></tr>
<tr><td>Other surface types: field is ignored</td></tr>
</table>
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>[4h, 1FFFCh]</td><td></td><td>in multiples of 4 (low 2 bits missing)</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>Software must ensure that this field is set to a value sufficiently large that array slices in the surface do not overlap. Refer to the Memory Data Formats section for information on how surfaces are stored.</td></tr>
</table>
</td>
</tr>
</table>

3DSTATE_DRAWING_RECTANGLE

3DSTATE_DRAWING_RECTANGLE

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The 3DSTATE_DRAWING_RECTANGLE command is used to set the 3D drawing rectangle and related state.

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: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 00h 3DSTATE_DRAWING_RECTANGLE Format: OpCode
	15:14	**Core Mode Select** Project: CHV, BSW Format: U2 Specifies which core this command will be considered valid and update based on the state in this command. Value / Name / Description / Project: 0h / Legacy / Both cores are enabled and will update the state. / All 1h / Core 0 Enabled / State will be updated in Core 0 only / All 2h / Core 1 Enabled / State will be updated in Core 1 only / All 3h / Reserved / / All
	13:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 2h Excludes DWord (0,1) Project: All Format: =n Total Length - 2
1	31:16	**Clipped Drawing Rectangle Y Min**

3DSTATE_DRAWING_RECTANGLE

DWord	Bit	Description
		Project: All Format: U16 in Pixels from Color Buffer origin (upper left corner) Specifies Ymin value of (inclusive) intersection of Drawing rectangle with the Color (Destination) Buffer, used for clipping. Pixels with Y coordinates less than Ymin will be clipped out. **Value** | **Name** [0,16383] | Device ignores bits 31:30 **Programming Notes** This value can be larger than Clipped Drawing Rectangle Y Max. If Ymin>Ymax, the clipped drawing rectangle is null, all polygons are discarded. If Ymin==Ymax, the clipped drawing rectangle is 1 pixel wide in the Y direction.
	15:0	**Clipped Drawing Rectangle X Min** Project: All Format: U16 in Pixels from Color Buffer origin (upper left corner) Specifies Xmin value of (inclusive) intersection of Drawing rectangle with the Color (Destination) Buffer, used for clipping. Pixels with X coordinates less than Xmin will be clipped out. **Value** | **Name** [0,16383] | Device ignores bits 15:14 **Programming Notes** This value can be larger than Clipped Drawing Rectangle X Max. If Xmin>Xmax, the clipped drawing rectangle is null, all polygons are discarded. If Xmin==Xmax, the clipped drawing rectangle is 1 pixel wide in the X direction.
2	31:16	**Clipped Drawing Rectangle Y Max** Project: All Format: U16 in Pixels from Color Buffer origin (upper left corner) Specifies Ymax value of (inclusive) intersection of Drawing rectangle with the Color (Destination) Buffer, used for clipping. Pixels with coordinates greater than Ymax will be clipped out. **Value** | **Name** [0,16383] | Device ignores bits 31:30 **Programming Notes** This value can be less than Clipped Drawing Rectangle Y Min. If Ymax<Ymin, the clipped drawing rectangle is null, all polygons are discarded. If Ymin==Ymax, the clipped drawing rectangle is 1 pixel wide in the Y direction.
	15:0	**Clipped Drawing Rectangle X Max** Project: All Format: U16 in Pixels from Color Buffer origin (upper left corner) Specifies Xmax value of (inclusive) intersection of Drawing rectangle with the Color (Destination) Buffer, used for clipping. Pixels with coordinates greater than Xmax will be clipped out.

3DSTATE_DRAWING_RECTANGLE

Value	Name	Project
[0,16383]	Device ignores bits 15:14	CHV, BSW

Programming Notes	Project
This value can be less than Clipped Drawing Rectangle X Min. If Xmax<Xmin, the clipped drawing rectangle is null, all polygons are discarded.If Xmin==Xmax, the clipped drawing rectangle is 1 pixel wide in the X direction.	CHV, BSW

3 — 31:16 — Drawing Rectangle Origin Y

Project:	All
Format:	S15 in Pixels from Color Buffer origin (upper left corner).

Description	Project
Range: [-16384,16383] (Bit 31 should be a sign extension)	CHV, BSW
Specifies Y origin of Drawing Rectangle (in whole pixels) relative to origin of the Color Buffer, used to map incoming (Draw Rectangle-relative) vertex positions to the Color Buffer space.	

15:0 — Drawing Rectangle Origin X

Project:	All
Format:	S15 in Pixels from Color Buffer origin (upper left corner).

Description	Project
Range: [-16384,16383] (Bit 15 should be a sign extension)	CHV, BSW
Specifies X origin of Drawing Rectangle (in whole pixels) relative to origin of the Color Buffer, used to map incoming (Draw Rectangle-relative) vertex positions to the Color Buffer space.	

3DSTATE_DS

3DSTATE_DS		
Project:	CHV, BSW	
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: 7h Excludes DWord (0,1) Project: CHV, BSW Format: =n Total Length - 2
1..2 **Project:** CHV, BSW	63:6	**Kernel Start Pointer** Project: CHV, BSW 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** Project: All Format: MBZ

3DSTATE_DS

3	31	**Single Domain Point Dispatch** Project: CHV, BSW Format: U1 Enumerated Type This field can be used to force single domain point SIMD4x2 DS threads. This field is ignored if **SIMD8 Dispatch Enable** is set.
	30	**Vector Mask Enable** Project: CHV, BSW 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.
	29:27	**Sampler Count** Project: CHV, BSW 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.

Single Domain Point Dispatch (bit 31)

Value	Name	Description
0h	Multiple	Dual domain point SIMD4x2 thread dispatches are allowed.
1h	Single	Single domain point SIMD4x2 thread dispatches are forced.

Workaround	Project
Workaround: The Single Domain Point Dispatch must always be set to 0.	CHV, BSW

Vector Mask Enable (bit 30)

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).

Sampler Count (bits 29:27)

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

3DSTATE_DS

		4h	13-16 Samplers — between 13 and 16 samplers used
	26	**Reserved**	
		Project:	CHV, BSW
		Format:	MBZ
	25:18	**Binding Table Entry Count**	
		Project:	CHV, BSW
		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.

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

17 — Thread Dispatch Priority

Project:	CHV, BSW
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

Project:	CHV, BSW
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

3DSTATE_DS

DWord	Bit	Description
	15	**Reserved** Project: CHV, BSW Format: MBZ
	14	**Accesses UAV** Project: CHV, BSW 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** Project: CHV, BSW 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.
	12:8	**Reserved** Project: CHV, BSW Format: MBZ
	7	**Software Exception Enable** Project: CHV, BSW 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** Project: CHV, BSW Format: MBZ
4..5 **Project:** CHV, BSW	63:10	**Scratch Space Base Pointer** Project: CHV, BSW 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.

3DSTATE_DS

DWord	Bit	Description
	9:4	**Reserved** Project: CHV, BSW Format: MBZ
	3:0	**Per-Thread Scratch Space** Project: CHV, BSW 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 **Project:** CHV, BSW	31:25	**Reserved** Project: CHV, BSW Format: MBZ
	24:20	**Dispatch GRF Start Register For URB Data** Project: CHV, BSW 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. **Value** / **Name** / **Description**: [0,31] — (none) — indicating GRF [R0, R31]
	19:18	**Reserved** Project: CHV, BSW Format: MBZ
	17:11	**Patch URB Entry Read Length** Project: CHV, BSW 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] — (none)

3DSTATE_DS

	10	**Reserved** Project: CHV, BSW Format: MBZ
	9:4	**Patch URB Entry Read Offset** Project: CHV, BSW 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: [0,63] — Name:
	3:0	**Reserved** Project: CHV, BSW Format: MBZ
7 **Project:** CHV, BSW	31	**Reserved** Project: CHV, BSW Format: MBZ
	30	**Reserved** Project: CHV, BSW Format: MBZ
	29:21	**Maximum Number of Threads** Project: CHV, BSW 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.
	20:11	**Reserved** Project: CHV, BSW Format: MBZ
	10	**Statistics Enable** Project: CHV, BSW

Maximum Number of Threads values:

Value	Name	Description	Project
[0,503]		indicating thread count of [1,504]	CHV, BSW
[0,79]		indicating thread count of [1,80]	CHV, BSW

3DSTATE_DS

		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** Project: CHV, BSW Format: MBZ
	4	**Reserved** Project: CHV, BSW Format: MBZ
	3	**SIMD8 Dispatch Enable** Project: CHV, BSW Format: Enable This field is used to specify how DS threads are dispatched. The setting of this field must agree with how the DS kernel was compiled. If ENABLED, SIMD8 DS thread dispatches are performed. The **Single Domain Point Dispatch** field is ignored. If DISABLED, SIMD4x2 thread dispatches are performed. The **Single Domain Point Dispatch** field can be used to force single domain point dispatches.
	2	**Compute W Coordinate Enable** Project: CHV, BSW 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** Project: CHV, BSW 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

DWord	Bit	Description
	0	**Function Enable** Project: CHV, BSW 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 **Project:** CHV, BSW	31:28	**Reserved** Project: CHV, BSW Format: MBZ
	27	**Reserved**
	26:21	**Vertex URB Entry Output Read Offset** Project: CHV, BSW Format: U6 Specifies the offset (in 256-bit units) at which Vertex URB data is to be read from the URB by SBE. Value: [0,63] — Name: (none)
	20:16	**Vertex URB Entry Output Length** Project: CHV, BSW Format: U5 Specifies the amount of URB data written for each Vertex URB entry, in 256-bit register increments. Value: [1,16] — Name: (none) **Programming Notes** This length does not include the vertex header.
	15:8	**User Clip Distance Clip Test Enable Bitmask** Project: CHV, BSW Format: Mask[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.

3DSTATE_DS		
	7:0	**User Clip Distance Cull Test Enable Bitmask** Project: CHV, BSW Format: Mask[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_DX9_CONSTANT_BUFFER_POOL_ALLOC

3DSTATE_DX9_CONSTANT_BUFFER_POOL_ALLOC

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command sets up the Gather Pool for Gather Buffers.

Programming Notes

This command must only be programmed when resource streamer is enabled thru batch buffer start and MI_RS_CONTROL has not disabled resource streamer.

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 GFXPIPE_3D
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 1Bh 3DSTATE_DX9_CONSTANT_BUFFER_POOL_ALLOC Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2 Value: 2h — Name: DWORD_COUNT_n **[Default]** — Project: CHV, BSW
1..2 **Project:** CHV, BSW	63:48	**Reserved** Project: CHV, BSW Format: MBZ
	47:13	**Dx9 Constant Buffer Pool Base Address** Project: CHV, BSW

3DSTATE_DX9_CONSTANT_BUFFER_POOL_ALLOC

		Format: GraphicsAddress[47:13]Dx9_Constant_Buffer_Pool Specifies the base address of the Dx9 Constant Buffer pool.
	12:11	**Reserved** Project: CHV, BSW Format: MBZ
	10	**Dx9 Constant Buffer Pool Enable** Project: CHV, BSW Format: Enable When this bit is set it enables HW Dx9 constants buffers. When this bit is cleared it disables HW Dx9 constant buffers, the local bits for the constant buffers are cleared and the buffers will not be save or restored as part of context.
	9:7	**Reserved** Project: CHV, BSW Format: MBZ
	6:0	**Surface Object Control State** Project: CHV, BSW Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this surface. **Programming Notes** Bit 2 is not programmable and is always zero.
3 **Project:** CHV, BSW	31:13	**Dx9 Constant Buffer Pool Buffer Size** Project: CHV, BSW Format: U19 This field specifies the size of the buffer in 8K pages. Any access which straddle or go past the end of the buffer will return 0. Note that BufferSize=0 indicates that there is no valid data in the buffer. **Restriction** Programming size of zero is illegal in the case that the pool is enabled.
	12:0	**Reserved** Project: CHV, BSW Format: MBZ

3DSTATE_DX9_CONSTANTB_PS

3DSTATE_DX9_CONSTANTB_PS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command sets a DX9 constant Boolean register for PS.

Programming Notes

- The 3DSTATE_DX9_CONSTANTB_PS is a variable length command.
- Programming this command in batch buffer requires that all float, integer and boolean constants initialized prior to any commands or events that cause the constants to be written to memory.

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 GFXPIPE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 3Eh 3DSTATE_DX9_CONSTANTB_PS Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Project: All Format: =n Total Length - 2

Value	Name
0h	**[Default]**
0h-10h	Excludes DWord (0,1)

3DSTATE_DX9_CONSTANTB_PS		
1	31:16	**Reserved** Project: All Format: MBZ
	15	**Global Constant Register** Project: All Format: U1 When this bit is set the global constant register set will be updated. When this bit is clear the local constant register set will be updated.
	14:4	**Reserved** Project: All Format: MBZ
	3:0	**Constant Register Index** Project: All Format: U4 This field specifies the index of 1st boolean to be updated.
2..n	31:0	**Entry** Format: DX9_CONSTANTB_ENTRY [CHV, BSW] The nth boolean to be updated.

3DSTATE_DX9_CONSTANTB_VS

3DSTATE_DX9_CONSTANTB_VS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command sets a DX9 constant Boolean register for PS.

Programming Notes

- The 3DSTATE_DX9_CONSTANTB_VS is a variable length command.
- Programming this command in batch buffer requires that all float, integer and boolean constants initialized prior to any commands or events that cause the constants to be written to memory.

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 GFXPIPE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 3Dh 3DSTATE_DX9_CONSTANTB_VS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2 Value / Name: 0h **[Default]** 0h-10h Excludes DWord (0,1)
1	31:16	**Reserved** Format: MBZ

3DSTATE_DX9_CONSTANTB_VS		
	15	**Global Constant Register** Format: U1 When this bit is set the global constant register set will be updated. When this bit is clear the local constant register set will be updated.
	14:4	**Reserved** Format: MBZ
	3:0	**Constant Register Index** Format: U4 This field specifies the index of 1st boolean to be updated.
2..n	31:0	**Entry** Format: DX9_CONSTANTB_ENTRY [CHV, BSW] The nth boolean to be updated.

3DSTATE_DX9_CONSTANTF_PS

3DSTATE_DX9_CONSTANTF_PS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command sets one or more DX9 constant float registers for PS.

Programming Notes

- The 3DSTATE_DX9_CONSTANTF_PS is a variable length command.
- Programming this command in batch buffer requires that all float, integer and boolean constants initialized prior to any commands or events that cause the constants to be written to memory.

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 GFXPIPE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 3Ah 3DSTATE_DX9_CONSTANTF_PS Format: OpCode
	15:11	**Reserved** Format: MBZ
	10:0	**DWord Length** Format: =n Total Length - 2 Value / Name: 1h: Excludes DWord (0,1) **[Default]** 1h-400h: multiples of 4
1	31:16	**Reserved** Format: MBZ

3DSTATE_DX9_CONSTANTF_PS		
	15	**Global Constant Register** Format: U1 When this bit is set the global constant register set will be updated. When this bit is clear the local constant register set will be updated.
	14:8	**Reserved** Format: MBZ
	7:0	**Constant Register Index** Format: U8 This field specifies the index of 1st 4 component float to be updated.
2..n	127:0	**Entry** Format: DX9_CONSTANTF_ENTRY [CHV, BSW] The four components of the nth float to be updated.

3DSTATE_DX9_CONSTANTF_VS

3DSTATE_DX9_CONSTANTF_VS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command sets one or more DX9 constant float registers for VS.

Programming Notes

- The 3DSTATE_DX9_CONSTANTF_VS is a variable length command.
- Programming this command in batch buffer requires that all float, integer and boolean constants initialized prior to any commands or events that cause the constants to be written to memory.

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 GFXPIPE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 39h 3DSTATE_DX9_CONSTANTF_VS Format: OpCode
	15:11	**Reserved** Format: MBZ
	10:0	**DWord Length** Format: =n Total Length - 2

Value	Name
1h	Excludes DWord (0,1) **[Default]**
1h-400h	multiples of 4

DWord	Bit	Description
1	31:16	**Reserved** Format: MBZ

3DSTATE_DX9_CONSTANTF_VS		
	15	**Global Constant Register** Format: U1 When this bit is set the global constant register set will be updated. When this bit is clear the local constant register set will be updated.
	14:8	**Reserved** Format: MBZ
	7:0	**Constant Register Index** Format: U8 This field specifies the index of 1st 4 component float to be updated.
2..n	127:0	**Entry** Format: DX9_CONSTANTF_ENTRY [CHV, BSW] The four components of the nth float to be updated.

3DSTATE_DX9_CONSTANTI_PS

3DSTATE_DX9_CONSTANTI_PS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command sets one or more DX9 constant integer registers for PS.

Programming Notes

- The 3DSTATE_DX9_CONSTANTI_PS is a variable length command.
- Programming this command in batch buffer requires that all float, integer and boolean constants initialized prior to any commands or events that cause the constants to be written to memory.

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 GFXPIPE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 3Ch 3DSTATE_DX9_CONSTANTI_PS Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2
1	31:16	**Reserved** Format: MBZ

DWord Length values:

Value	Name	Description	Project
1h	**[Default]**	Excludes DWord (0,1)	CHV, BSW
0h-80h	multiples of 4		CHV, BSW

3DSTATE_DX9_CONSTANTI_PS		
	15	**Global Constant Register** Project: All Format: U1 When this bit is set the global constant register set will be updated. When this bit is clear the local constant register set will be updated.
	14:5	**Reserved** Project: All Format: MBZ
	4:0	**Constant Register Index** Project: All Format: U5 This field specifies the index of 1st 4 component integer to be updated.
2..n	127:0	**Entry** Format: DX9_CONSTANTI_ENTRY [CHV, BSW] The four components of the nth float to be updated.

3DSTATE_DX9_CONSTANTI_VS

3DSTATE_DX9_CONSTANTI_VS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command sets one or more DX9 constant integer registers for PS.

Programming Notes

- The 3DSTATE_DX9_CONSTANTI_VS is a variable length command.
- Programming this command in batch buffer requires that all float, integer and boolean constants initialized prior to any commands or events that cause the constants to be written to memory.

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 GFXPIPE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 3Bh 3DSTATE_DX9_CONSTANTI_VS Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2
1	31:16	**Reserved** Format: MBZ

DWord Length values:

Value	Name	Description	Project
0h	**[Default]**	Excludes DWord (0,1)	CHV, BSW
0h-80h	multiples of 4		CHV, BSW

3DSTATE_DX9_CONSTANTI_VS		
	15	**Global Constant Register** Project: All Format: U1 When this bit is set the global constant register set will be updated. When this bit is clear the local constant register set will be updated.
	14:5	**Reserved** Project: All Format: MBZ
	4:0	**Constant Register Index** Project: All Format: U5 This field specifies the index of 1st 4 component integer to be updated.
2..n	127:0	**Entry** Format: DX9_CONSTANTI_ENTRY [CHV, BSW] The four components of the nth float to be updated.

3DSTATE_DX9_GENERATE_ACTIVE_PS

3DSTATE_DX9_GENERATE_ACTIVE_PS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The 3DSTATE_DX9_GENERATE_ACTIVE_PS command is used to generate fixed functions' DX9 Constant Buffer. A DX9 Constant register is made active by writing it out to the constant buffer.

Programming Notes

Restriction: The global and local buffers are not initialized after reset. Any data written without being initialized will be undefined. DX9 constant buffers are written due to context save/restore or the Generate Active Command.

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: 42h 3DSTATE_DX9_GENERATE_ACTIVE_PS Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**Dword Length** Default Value: 0h Excludes Dword (0,1) Project: All Format: =n Total Length - 2
1	31:24	**Reserved** Project: All Format: MBZ

3DSTATE_DX9_GENERATE_ACTIVE_PS

	23:13	**Pointer to PS Constant Buffer** Project: All Format: ConstantBufferOffset[23:13]BINDING_TABLE_STATE* Specifies the 8KB aligned address offset of the PS function's Dx9 constant buffer. This offset is relative to the DX9 Constant buffer Base Address.
	12	**DX9 Enable** Project: All Format: Enable Format: U1 When this bit is set, the Resource Streamer will generate the PS constant buffer according to the DX9 rules: 1. Valid local register are made active. 2. Global register becomes active, unless the corresponding local register is valid. 3. Local register valids are reset. When this bit is cleared, the Resource Streamer will generate the PS constant buffer according to the DX8 rules: 1. Global register become active. 2. Local register valids are reset. **Programming Notes** In DX8 mode software will set all constants as globals, even ones locally defined within a shader.
	11	**Clamp Enable** Project: All Format: Enable Format: U1 When this bit is set, the Resource Streamer will generate the PS constant buffer with the global values clamped to [-1,1]. When this bit is cleared, the Resource Streamer will generate the PS constant buffer without the global value clamped. **Programming Notes** / **Project** The clamping only affects the values written out to the constant buffer and not the on-die registers. — CHV, BSW
	10:8	**Reserved** Project: CHV, BSW Format: MBZ
	7:0	**Reserved** Project: All Format: MBZ

3DSTATE_DX9_GENERATE_ACTIVE_VS

3DSTATE_DX9_GENERATE_ACTIVE_VS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The 3DSTATE_DX9_GENERATE_ACTIVE_VS command is used to generate fixed functions' DX9 Constant Buffer. A DX9 Constant register is made active by writing it out to the constant buffer.

Programming Restriction:The global and local buffers are not initialized after reset. Any data written without being initialized will be undefined. DX9 constant buffers are written due to context save/restore or the Generate Active Command.

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: 41h 3DSTATE_DX9_GENERATE_ACTIVE_VS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:24	**Reserved** Format: MBZ
	23:13	**Pointer to VS Constant Buffer** Format: ConstantBufferOffset[23:13] Specifies the 8KB aligned address offset of the VS function's Dx9 constant buffer. This offset is relative to the DX9 Constant buffer Base Address.

3DSTATE_DX9_GENERATE_ACTIVE_VS

	12	**DX9 Enable** Format: Enable When this bit is set, the Resource Streamer will generate the VS constant buffer according to the DX9 rules: 1. Valid local register are made active. 2. Global register becomes active, unless the corresponding local register is valid. 3. Local register valids are reset. When this bit is cleared, the Resource Streamer will generate the VS constant buffer according to the DX8 rules: 1. Global register become active. 2. Local register valids are reset. **Programming Notes** In DX8 mode software will set all constants as globals, even ones locally defined within a shader.
	11	**Clamp Enable** Format: Enable When this bit is set, the Resource Streamer will generate the VS constant buffer with the global values clamped to [-1,1]. When this bit is cleared, the Resource Streamer will generate the VS constant buffer without the global value clamped. **Programming Notes** / **Project** The clamping only affects the values written out to the constant buffer and not the on-die registers. / CHV, BSW
	10:8	**Reserved** Project: CHV, BSW Format: MBZ
	7:0	**Reserved** Format: MBZ

3DSTATE_DX9_LOCAL_VALID_PS

3DSTATE_DX9_LOCAL_VALID_PS		
Project:	CHV, BSW	
Source:	RenderCS	
Length Bias:	2	
This command sets the local valid bits for the DX9 Constant Buffer		
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 GFXPIPE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 40h 3DSTATE_DX9_LOCAL_VALID_PS Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**Dword Length** Default Value: 9h Excludes Dword (0,1) Project: CHV, BSW Format: =n Total Length - 2
1..8	31:0	**Local ConstantF Valid Bits** Project: All Format: U32 Each bit field when set indicates that the corresponding local register is valid. When the bit is clear it indicates the local register is invalid.
9	31:0	**Local ConstantI Valid Bits** Project: CHV, BSW Format: U32 Each bit field when set indicates that the corresponding local register is valid. When the bit is clear it indicates the local register is invalid.

3DSTATE_DX9_LOCAL_VALID_PS		
10	31:16	**Reserved** Project: All Format: MBZ
	15:0	**Local ConstantB Valid Bits** Project: CHV, BSW Format: U16 Each bit field when set indicates that the corresponding local register is valid. When the bit is clear it indicates the local register is invalid.

3DSTATE_DX9_LOCAL_VALID_VS

3DSTATE_DX9_LOCAL_VALID_VS		
Project: CHV, BSW Source: RenderCS Length Bias: 2		
This command sets the local valid bits for the DX9 Constant Buffer		
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 GFXPIPE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 3Fh 3DSTATE_DX9_LOCAL_VALID_VS Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 9h Excludes DWord (0,1) Project: CHV, BSW Format: =n Total Length - 2
1..8	31:0	**Local ConstantF Valid Bits** Project: All Format: U32 Each bit field when set indicates that the corresponding local register is valid. When the bit is clear it indicates the local register is invalid.
9	31:0	**Local ConstantI Valid Bits** Project: CHV, BSW Format: U32 Each bit field when set indicates that the corresponding local register is valid. When the bit is clear it indicates the local register is invalid.

3DSTATE_DX9_LOCAL_VALID_VS		
10 **Project:** CHV, BSW	31:16	**Reserved** Project: CHV, BSW Format: MBZ
	15:0	**Local ConstantB Valid Bits** Project: CHV, BSW Format: U16 Each bit field when set indicates that the corresponding local register is valid. When the bit is clear it indicates the local register is invalid.

3DSTATE_GATHER_CONSTANT_DS

3DSTATE_GATHER_CONSTANT_DS	
Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command uses the constant buffer binding table entries to reference constant buffer surface states for the DS unit. The constant data in these is gathered and packed according to a gather table contained in this command.

Programming Notes
The HW generated binding table must be enabled to use this command.
The constant buffer block (group of aligned 16 binding table entries) must be set before this command is issued.
If the surface type is NULL, any fetch using the surface state base address is not bound by the size of the surface state and the fetch still occurs.
The length of the gather table is derived from the total length of the command. The command length is in DWords, but the gather table entries are 16 bits in length. If there is an unused odd entry at the end of the command the channel mask should be set to all 0s.
When a 3DSTATE_GATHER_CONSTANT_* command is used there must be a matching 3DSTATE_CONSTANT_*. Furthermore the 3DSTATE_CONSTANT_* must occur in the same order as the 3DSTATE_GATHER_CONSTANT_*. For example if a 3DSTATE_GATHER_CONSTANT_VS occurs before a 3DSTATE_GATHER_CONSTANT_PS, then the 3DSTATE_CONSTANT_VS must occur before the 3DSTATE_CONSTANT_PS.
If Gather pool is enabled, there must be a corresponding 3DSTATE_GATHER_CONSTANT command with any 3DSTATE_CONSTANT for any particular shader. To avoid any update to the Gather pool, and yet program the 3DSTATE_CONSTANT for a particular shader, send a 3DSTATE_GATHER_CONSTANT command with all valid bits set to zero.
The following commands must be executed after any 3DSTATE_GATHER_CONSTANT_* command that has Constant Buffer Valid greater than zero: •(N times, mininum number is 4) MI_RS_STORE_DATA_IMM –To force engine idle before executing the next instruction. Write must occur to address that will not corrupt memory: •Resource Streamer Flush = 1 •3DSTATE_GATHER_CONSTANT_* (Ensures correct timing of sync between resource streamer and render pipeline) •The Constant Buffer Valid field should be zero and the Dword length equal to 1h. •3DSTATE_CONSTANT_*: •All values match the previous 3DSTATE_CONSTANT_*

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

3DSTATE_GATHER_CONSTANT_DS		
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 37h 3DSTATE_GATHER_CONSTANT_DS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2 Value / Name / Description: 1 / DWORD_COUNT_n **[Default]** / excludes DWords 0,1 [1,128] / Range / 1-128 Entries
1	31:16	**Constant Buffer Valid** Format: U16 This field specifies which of the 16 constant buffers are used in the push constant gather. If a bit is set it indicates the corresponding constant buffer is used. If a bit is clear it indicates the corresponding constant buffer is not used. If this field is zero it indicate that the gather buffer is not used.
	15:12	**Constant Buffer Binding Table Block** Format: U4 This field specifies the 16 entry block constant buffer in the binding table. The constant buffer entry block must be aligned on a 16 entry boundary.
	11:2	**Reserved** Format: MBZ
	1	**Reserved** Project: CHV, BSW Format: MBZ
	0	**Reserved** Format: MBZ
2	31:23	**Reserved** Format: MBZ
	22:6	**Gather Buffer Offset** Format: GatherBufferOffset[22:6] This field specifies the offset of the gather buffer within the Gather Pool

3DSTATE_GATHER_CONSTANT_DS

		Programming Notes SW increments the offset by the size of the gather buffer in 512 bit units for each gather buffer generated.
	5	**Constant Buffer Dx9 Generate Stall** Project: CHV, BSW Format: Enable When this bit is set the resource streamer will wait for the Dx9 constant buffer generator to be done before issuing this command to ensure buffer synchronization.
	4	**Reserved** Project: All Format: MBZ
	3	**Reserved** Project: CHV, BSW Format: MBZ
	2:0	**Reserved** Project: All Format: MBZ
3..n	15:0	**Entry** Format: GATHER_CONSTANT_ENTRY [CHV, BSW]

3DSTATE_GATHER_CONSTANT_GS

3DSTATE_GATHER_CONSTANT_GS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command uses the constant buffer binding table entries to reference constant buffer surface states for GS unit. The constant data in these is gathered and packed according to a gather table contained in this command.

Programming Notes
The HW generated binding table must be enabled to use this command.
The constant buffer block (group of aligned 16 binding table entries) must be set before this command is issued.
If the surface type is NULL, any fetch using the surface state base address is not bound by the size of the surface state and the fetch still occurs.
The length of the gather table is derived from the total length of the command. The command length is in DWords, but the gather table entries are 16 bits in length. If there is an unused odd entry at the end of the command the channel mask should be set to all 0s.
When a 3DSTATE_GATHER_CONSTANT_* command is used there must be a matching 3DSTATE_CONSTANT_*. Furthermore the 3DSTATE_CONSTANT_* must occur in the same order as the 3DSTATE_GATHER_CONSTANT_*. For example if a 3DSTATE_GATHER_CONSTANT_VS occurs before a 3DSTATE_GATHER_CONSTANT_PS, then the 3DSTATE_CONSTANT_VS must occur before the 3DSTATE_CONSTANT_PS.
If Gather pool is enabled, there must be a corresponding 3DSTATE_GATHER_CONSTANT command with any 3DSTATE_CONSTANT for any particular shader. To avoid any update to the Gather pool, and yet program the 3DSTATE_CONSTANT for a particular shader, send a 3DSTATE_GATHER_CONSTANT command with all valid bits set to zero.
The following commands must be executed after any 3DSTATE_GATHER_CONSTANT_* command that has Constant Buffer Valid greater than zero: •(N times, mininum number is 4) MI_RS_STORE_DATA_IMM –To force engine idle before executing the next instruction. Write must occur to address that will not corrupt memory: •Resource Streamer Flush = 1 •3DSTATE_GATHER_CONSTANT_* (Ensures correct timing of sync between resource streamer and render pipeline) •The Constant Buffer Valid field should be zero and the Dword length equal to 1h. •3DSTATE_CONSTANT_*: •All values match the previous 3DSTATE_CONSTANT_*

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

3DSTATE_GATHER_CONSTANT_GS

DWord	Bit	Description
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 35h 3DSTATE_GATHER_CONSTANT_GS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2 Value: 1 — Name: DWORD_COUNT_n **[Default]** — Description: excludes DWords 0,1 Value: [1,128] — Name: Range — Description: 1-128 Entries
1	31:16	**Constant Buffer Valid** Format: U16 This field specifies which of the 16 constant buffers are used in the push constant gather. If a bit is set it indicates the corresponding constant buffer is used. If a bit is clear it indicates the corresponding constant buffer is not used. If this field is zero it indicate that the gather buffer is not used.
	15:12	**Constant Buffer Binding Table Block** Format: U4 This field specifies the 16 entry block constant buffer in the binding table. The constant buffer entry block must be aligned on a 16 entry boundary.
	11:2	**Reserved** Format: MBZ
	1	**Reserved** Project: CHV, BSW Format: MBZ
	0	**Reserved** Format: MBZ
2	31:23	**Reserved** Format: MBZ
	22:6	**Gather Buffer Offset** Format: GatherBufferOffset[22:6] This field specifies the offset of the gather buffer within the Gather Pool.

3DSTATE_GATHER_CONSTANT_GS		
		Programming Notes SW increments the offset by the size of the gather buffer in 512 bit units for each gather buffer generated.
	5	**Constant Buffer Dx9 Generate Stall** Project: CHV, BSW Format: Enable When this bit is set the resource streamer will wait for the Dx9 constant buffer generator to be done before issuing this command to ensure buffer synchronization.
	4	**Reserved** Project: All Format: MBZ
	3	**Reserved** Project: CHV, BSW Format: MBZ
	2:0	**Reserved** Project: All Format: MBZ
3..n	15:0	**Entry** Format: GATHER_CONSTANT_ENTRY [CHV, BSW]

3DSTATE_GATHER_CONSTANT_HS

3DSTATE_GATHER_CONSTANT_HS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command uses the constant buffer binding table entries to reference constant buffer surface states for HS unit. The constant data in these is gathered and packed according to a gather table contained in this command.

Programming Notes
The HW generated binding table must be enabled to use this command.
The constant buffer block (group of aligned 16 binding table entries) must be set before this command is issued.
If the surface type is NULL, any fetch using the surface state base address is not bound by the size of the surface state and the fetch still occurs.
The length of the gather table is derived from the total length of the command. The command length is in DWords, but the gather table entries are 16 bits in length. If there is an unused odd entry at the end of the command the channel mask should be set to all 0s.
When a 3DSTATE_GATHER_CONSTANT_* command is used there must be a matching 3DSTATE_CONSTANT_*. Furthermore the 3DSTATE_CONSTANT_* must occur in the same order as the 3DSTATE_GATHER_CONSTANT_*. For example if a 3DSTATE_GATHER_CONSTANT_VS occurs before a 3DSTATE_GATHER_CONSTANT_PS, then the 3DSTATE_CONSTANT_VS must occur before the 3DSTATE_CONSTANT_PS.
If Gather pool is enabled, there must be a corresponding 3DSTATE_GATHER_CONSTANT command with any 3DSTATE_CONSTANT for any particular shader. To avoid any update to the Gather pool, and yet program the 3DSTATE_CONSTANT for a particular shader, send a 3DSTATE_GATHER_CONSTANT command with all valid bits set to zero.
The following commands must be executed after any 3DSTATE_GATHER_CONSTANT_* command that has Constant Buffer Valid greater than zero: •(N times, mininum number is 4) MI_RS_STORE_DATA_IMM –To force engine idle before executing the next instruction. Write must occur to address that will not corrupt memory: •Resource Streamer Flush = 1 •3DSTATE_GATHER_CONSTANT_* (Ensures correct timing of sync between resource streamer and render pipeline) •The Constant Buffer Valid field should be zero and the Dword length equal to 1h. •3DSTATE_CONSTANT_*: •All values match the previous 3DSTATE_CONSTANT_*

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

3DSTATE_GATHER_CONSTANT_HS

DWord	Bit	Description
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 36h 3DSTATE_GATHER_CONSTANT_HS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2

Value	Name	Description
1	DWORD_COUNT_n **[Default]**	excludes DWords 0,1
[1,128]	Range	1-128 Entries

DWord	Bit	Description
1	31:16	**Constant Buffer Valid** Format: U16 This field specifies which of the 16 constant buffers are used in the push constant gather. If a bit is set it indicates the corresponding constant buffer is used. If a bit is clear it indicates the corresponding constant buffer is not used. If this field is zero it indicate that the gather buffer is not used.
	15:12	**Constant Buffer Binding Table Block** Format: U4 This field specifies the 16 entry block constant buffer in the binding table. The constant buffer entry block must be aligned on a 16 entry boundary.
	11:2	**Reserved** Format: MBZ
	1	**Reserved** Project: CHV, BSW Format: MBZ
	0	**Reserved** Format: MBZ
2	31:23	**Reserved** Format: MBZ
	22:6	**Gather Buffer Offset** Format: GatherBufferOffset[22:6] This field specifies the offset of the gather buffer within the Gather Pool.

3DSTATE_GATHER_CONSTANT_HS

		Programming Notes SW increments the offset by the size of the gather buffer in 512 bit units for each gather buffer generated.
	5	**Constant Buffer Dx9 Generate Stall** Project: CHV, BSW Format: Enable When this bit is set the resource streamer will wait for the Dx9 constant buffer generator to be done before issuing this command to ensure buffer synchronization.
	4	**Reserved** Project: All Format: MBZ
	3	**Reserved** Project: CHV, BSW Format: MBZ
	2:0	**Reserved** Project: All Format: MBZ
3..n	15:0	**Entry** Format: GATHER_CONSTANT_ENTRY [CHV, BSW]

3DSTATE_GATHER_CONSTANT_PS

3DSTATE_GATHER_CONSTANT_PS	
Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command uses the constant buffer binding table entries to reference constant buffer surface states for PS unit. The constant data in these is gathered and packed according to a gather table contained in this command.

Programming Notes
The HW generated binding table must be enabled to use this command.
The constant buffer block (group of aligned 16 binding table entries) must be set before this command is issued.
If the surface type is NULL, any fetch using the surface state base address is not bound by the size of the surface state and the fetch still occurs.
The length of the gather table is derived from the total length of the command. The command length is in DWords, but the gather table entries are 16 bits in length. If there is an unused odd entry at the end of the command the channel mask should be set to all 0s.
When a 3DSTATE_GATHER_CONSTANT_* command is used there must be a matching 3DSTATE_CONSTANT_*. Furthermore the 3DSTATE_CONSTANT_* must occur in the same order as the 3DSTATE_GATHER_CONSTANT_*. For example if a 3DSTATE_GATHER_CONSTANT_VS occurs before a 3DSTATE_GATHER_CONSTANT_PS, then the 3DSTATE_CONSTANT_VS must occur before the 3DSTATE_CONSTANT_PS.
If Gather pool is enabled, there must be a corresponding 3DSTATE_GATHER_CONSTANT command with any 3DSTATE_CONSTANT for any particular shader. To avoid any update to the Gather pool, and yet program the 3DSTATE_CONSTANT for a particular shader, send a 3DSTATE_GATHER_CONSTANT command with all valid bits set to zero.
The following commands must be executed after any 3DSTATE_GATHER_CONSTANT_* command that has Constant Buffer Valid greater than zero: •(N times, mininum number is 4) MI_RS_STORE_DATA_IMM –To force engine idle before executing the next instruction. Write must occur to address that will not corrupt memory: •Resource Streamer Flush = 1 •3DSTATE_GATHER_CONSTANT_* (Ensures correct timing of sync between resource streamer and render pipeline) •The Constant Buffer Valid field should be zero and the Dword length equal to 1h. •3DSTATE_CONSTANT_*: •All values match the previous 3DSTATE_CONSTANT_*

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

3DSTATE_GATHER_CONSTANT_PS

DWord	Bit	Description
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 38h 3DSTATE_GATHER_CONSTANT_PS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2
1	31:16	**Constant Buffer Valid** Format: U16 This field specifies which of the 16 constant buffers are used in the push constant gather. If a bit is set it indicates the corresponding constant buffer is used. If a bit is clear it indicates the corresponding constant buffer is not used. If this field is zero it indicate that the gather buffer is not used.
	15:12	**Constant Buffer Binding Table Block** Format: U4 This field specifies the 16 entry block constant buffer in the binding table. The constant buffer entry block must be aligned on a 16 entry boundary.
	11:2	**Reserved** Format: MBZ
	1:0	**Reserved** Project: CHV, BSW Format: MBZ
2	31:23	**Reserved** Format: MBZ
	22:6	**Gather Buffer Offset** Format: GatherBufferOffset[22:6] This field specifies the offset of the gather buffer within the Gather Pool.

DWord Length values (bits 7:0):

Value	Name	Description
1	DWORD_COUNT_n **[Default]**	excludes DWords 0,1
[1,128]	Range	1-128 Entries

Gather Buffer Offset (bits 22:6):

Programming Notes
SW increments the offset by the size of the gather buffer in 512 bit units for each gather buffer generated.

3DSTATE_GATHER_CONSTANT_PS		
	5	**Constant Buffer Dx9 Generate Stall** Project: CHV, BSW Format: Enable When this bit is set the resource streamer will wait for the Dx9 constant buffer generator to be done before issuing this command to ensure buffer synchronization.
	4	**Constant Buffer Dx9 Enable** Format: Enable When this bit is set it indicates that the constant buffer is a HW generated Dx9 constant buffer. The resource streamer will wait for the Dx9 constant buffer generator to be done before issuing this command to ensure buffer synchronization. Additionally the Dx9 constant buffers are a single buffer but larger than 4KB. Internally the HW will treat the DX9 buffer as 2 constant buffers. When this bit is enable only the 1st constant buffer valid bit is set. The 2nd constant buffer surface pointer will automatically be the 1st pointer + 4KB.
	3	**Reserved** Project: CHV, BSW Format: MBZ
	2:0	**Reserved** Format: MBZ
3..n	15:0	**Entry** Format: GATHER_CONSTANT_ENTRY [CHV, BSW]

3DSTATE_GATHER_CONSTANT_VS

<table>
<tr><th colspan="3">3DSTATE_GATHER_CONSTANT_VS</th></tr>
<tr><td colspan="3">Project: CHV, BSW
Source: RenderCS
Length Bias: 2</td></tr>
<tr><td colspan="3">This command uses the constant buffer binding table entries to reference constant buffer surface states for VS unit. The constant data in these is gathered and packed according to a gather table contained in this command.</td></tr>
<tr><td colspan="3">Programming Notes</td></tr>
<tr><td colspan="3">The HW generated binding table must be enabled to use this command.</td></tr>
<tr><td colspan="3">The constant buffer block (group of aligned 16 binding table entries) must be set before this command is issued.</td></tr>
<tr><td colspan="3">If the surface type is NULL, any fetch using the surface state base address is not bound by the size of the surface state and the fetch still occurs.</td></tr>
<tr><td colspan="3">The length of the gather table is derived from the total length of the command. The command length is in DWords, but the gather table entries are 16 bits in length. If there is an unused odd entry at the end of the command the channel mask should be set to all 0s.</td></tr>
<tr><td colspan="3">When a 3DSTATE_GATHER_CONSTANT_* command is used there must be a matching 3DSTATE_CONSTANT_*. Furthermore the 3DSTATE_CONSTANT_* must occur in the same order as the 3DSTATE_GATHER_CONSTANT_*. For example if a 3DSTATE_GATHER_CONSTANT_VS occurs before a 3DSTATE_GATHER_CONSTANT_PS, then the 3DSTATE_CONSTANT_VS must occur before the 3DSTATE_CONSTANT_PS.</td></tr>
<tr><td colspan="3">If Gather pool is enabled, there must be a corresponding 3DSTATE_GATHER_CONSTANT command with any 3DSTATE_CONSTANT for any particular shader. To avoid any update to the Gather pool, and yet program the 3DSTATE_CONSTANT for a particular shader, send a 3DSTATE_GATHER_CONSTANT command with all valid bits set to zero.</td></tr>
<tr><td colspan="3">The following commands must be executed after any 3DSTATE_GATHER_CONSTANT_* command that has Constant Buffer Valid greater than zero: •(N times, mininum number is 4) MI_RS_STORE_DATA_IMM –To force engine idle before executing the next instruction. Write must occur to address that will not corrupt memory:
•Resource Streamer Flush = 1
•3DSTATE_GATHER_CONSTANT_* (Ensures correct timing of sync between resource streamer and render pipeline) •The Constant Buffer Valid field should be zero and the Dword length equal to 1h.
•3DSTATE_CONSTANT_*: •All values match the previous 3DSTATE_CONSTANT_*</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="2">0</td><td>31:29</td><td>Command Type
Default Value: 3h GFXPIPE
Format: OpCode</td></tr>
<tr><td>28:27</td><td>Command SubType
Default Value: 3h GFXPIPE_3D
Format: OpCode</td></tr>
</table>

3DSTATE_GATHER_CONSTANT_VS

DWord	Bit	Description
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 34h 3DSTATE_GATHER_CONSTANT_VS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2 Value: 0 — Name: DWORD_COUNT_n **[Default]** — Description: excludes DWords 0,1 Value: [1,128] — Name: Range — Description: 1-128 Entries
1	31:16	**Constant Buffer Valid** Format: U16 This field specifies which of the 16 constant buffers are used in the push constant gather. If a bit is set it indicates the corresponding constant buffer is used. If a bit is clear it indicates the corresponding constant buffer is not used. If this field is zero it indicate that the gather buffer is not used.
	15:12	**Constant Buffer Binding Table Block** Format: U4 This field specifies the 16 entry block constant buffer in the binding table. The constant buffer entry block must be aligned on a 16 entry boundary.
	11:2	**Reserved** Format: MBZ
	1:0	**Reserved** Project: CHV, BSW Format: MBZ
2	31:23	**Reserved** Format: MBZ
	22:6	**Gather Buffer Offset** Format: GatherBufferOffset[22:6] This field specifies the offset of the gather buffer within the Gather Pool. **Programming Notes** SW increments the offset by the size of the gather buffer in 512 bit units for each gather buffer generated.

3DSTATE_GATHER_CONSTANT_VS

	5	**Constant Buffer Dx9 Generate Stall** Project: CHV, BSW Format: Enable When this bit is set the resource streamer will wait for the Dx9 constant buffer generator to be done before issuing this command to ensure buffer synchronization.
	4	**Constant Buffer Dx9 Enable** Format: Enable When this bit is set it indicates that the constant buffer is a HW generated Dx9 constant buffer. The resource streamer will wait for the Dx9 constant buffer generator to be done before issuing this command to ensure buffer synchronization. Additionally the Dx9 constant buffers are a single buffer but larger than 4KB. Internally the HW will treat the DX9 buffer as 2 constant buffers. When this bit is enable only the 1st constant buffer valid bit is set. The 2nd constant buffer surface pointer will automatically be the 1st pointer + 4KB.
	3	**Reserved** Project: CHV, BSW Format: MBZ
	2:0	**Reserved** Format: MBZ
3..n	15:0	**Entry** Format: GATHER_CONSTANT_ENTRY [CHV, BSW]

3DSTATE_GATHER_POOL_ALLOC

3DSTATE_GATHER_POOL_ALLOC

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command sets up the Gather Pool for Gather Buffers.

Programming Notes
If the Gather Constant Pool is enabled and RS is enabled, then for each 3DSTATE_CONSTANT_* command there must be a corresponding 3DSTATE_GATHER_CONSTANT_* command. If gather pool is enabled, then Buffer 1 of the 3DSTATE_CONSTANT command address will be an offset from the Gather Pool Base Address.
The gather constants can only be enabled if the binding table generator is also enabled. This command must only be programmed when resource streamer is enabled thru batch buffer start and MI_RS_CONTROL has not disabled resource streamer.

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: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 1Ah 3DSTATE_GATHER_POOL_ALLOC Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Value: 2h — Name: DWORD_COUNT_n **[Default]** — Project: CHV, BSW
1..2 **Project:** CHV, BSW	63:12	**Gather Pool Base Address** Project: CHV, BSW Format: GraphicsAddress[63:12]Gather_Pool

<table>
<tr><th colspan="3">3DSTATE_GATHER_POOL_ALLOC</th></tr>
<tr><td></td><td></td><td>GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].</td></tr>
<tr><td></td><td>11</td><td>Gather Pool Enable
Project: CHV, BSW
Format: Enable
When this bit is set it enables HW gathering of push constants. When this bit is cleared it disables HW gathering of push constants.</td></tr>
<tr><td></td><td>10:7</td><td>Reserved
Project: CHV, BSW
Format: MBZ</td></tr>
<tr><td></td><td>6:0</td><td>Memory Object Control State
Project: CHV, BSW
Format: MEMORY_OBJECT_CONTROL_STATE
Specifies the memory object control state for this surface.
Programming Notes
Bit 2 is not programmable and is always zero.</td></tr>
<tr><td>3
Project:
CHV, BSW</td><td>31:12</td><td>Gather Pool Buffer Size
Project: CHV, BSW
Format: U20
This field specifies the size of the buffer in 4K pages. Any access which straddle or go past the end of the buffer will return undefined data. Note that BufferSize=0 indicates that there is no valid data in the buffer.
<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,1048575]</td><td></td></tr></table>
Restriction
Programming size of zero is illegal in the case that the pool is enabled.</td></tr>
<tr><td></td><td>11:0</td><td>Reserved
Project: CHV, BSW
Format: MBZ</td></tr>
</table>

3DSTATE_GS

3DSTATE_GS		
Project: CHV, BSW 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** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 8h Excludes DWord (0,1) Format: =n
1..2	63:6	**Kernel Start Pointer** Project: All 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** Project: All Format: MBZ
3	31	**Single Program Flow**

3DSTATE_GS

		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: U1 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**
	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**

3DSTATE_GS

		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** Project: All Specifies the initial floating point mode used by the dispatched thread. Value / Name / Description / Project: 0h / IEEE-754 / Use IEEE-754 Rules / All 1h / Alternate / Use alternate rules / All
	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.*
	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.*

3DSTATE_GS

DWord	Bit	Description
	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: [1,32] — Name: (none)
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: [0,11] — Name: (none) — Description: indicating [1K Bytes, 2M Bytes] Programming Notes: Temporary w/a: If Per-Thread Scratch space is greater than 0, GS Number of URB Entries must be less than or equal to the GS Maximum Number of Threads. — Project: CHV, BSW
6	31	**Reserved** Project: All Format: MBZ
	30:29	**Reserved** Project: CHV, BSW Format: MBZ

3DSTATE_GS

	28:23	**Output Vertex Size** Project: All 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** Project: All Format: 3DPrimType This field specifies the topology type (3DPrimType) to be associated with GS-thread output vertices (if any).
	16:11	**Vertex URB Entry Read Length** Project: All 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** Project: All 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.
	9:4	**Vertex URB Entry Read Offset** Project: All 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.

3DSTATE_GS

DWord	Bit	Description
	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.
7	31:24	**Maximum Number of Threads** Format: U8/2-1 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.
	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 maixumum number of vertices output by the GS thread. It is UNDEFINED for a GS thread to report more output vertices than can be accomodated in a non-zero-sized header. (If the header size is zero, by definition neither cut nor StreamID bits are defined.

Dispatch GRF Start Register For URB Data (3:0)

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

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
For simd8: For InstanceCount == 1: numVerticesPerObject + 2 For InstanceCount > 1: 3

Maximum Number of Threads (31:24)

Value	Name	Description
[3,251]		indicating thread count of [8,504]
[3,39]		indicating thread count of [8,80]

Programming Notes
Note that this "Maximum Number of Threads" field is different from the other FF stages in that only an even number of threads.

Control Data Header Size (23:20)

Value	Name
[0,8]	32B Units

3DSTATE_GS

	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.
	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.

Instance Control values:

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

Dispatch Mode values:

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 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.	Not valid for objects with more than 6 vertices per object.

3DSTATE_GS

		Programming Notes The GS must be allocated at least two URB handles or behavior is UNDEFINED for Dual Instance or Dual Object mode. At least 8 URB entries must be allocated in order to use SIMD8 DispatchMode.
	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.
	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.

3DSTATE_GS

	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.

Programming Notes
Workaround: reorder mode must be set to REORDER_LEADING and reordering must be done in the Geometry shader.
Workaround
Workaround: To work around a CHV, BSW issue, reorder mode must be set to REORDER_LEADING when GS is disabled.

	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.

3DSTATE_GS		
	0	**Enable** Format: Enable Specifies whether the GS stage is enabled or disabled (pass-through).
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	**Reserved** Format: MBZ
9	31:28	**Reserved** Format: MBZ
	27	**Reserved**
	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] |

3DSTATE_GS

	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_HIER_DEPTH_BUFFER

3DSTATE_HIER_DEPTH_BUFFER

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

Description	Project
This command sets the surface state of the hierarchical depth buffer, delivered as a pipelined state command. However, the state change pipelining isn't completely transparent (see restriction below).	
WM HW will internally manage the draining pipe and flushing of the caches when this command is issued. The PIPE_CONTROL restrictions are removed.	CHV, BSW

Programming Notes
Restriction: Prior to changing Depth/Stencil Buffer state (i.e., any combination of 3DSTATE_DEPTH_BUFFER, 3DSTATE_CLEAR_PARAMS, 3DSTATE_STENCIL_BUFFER, 3DSTATE_HIER_DEPTH_BUFFER) SW must first issue a pipelined depth stall (PIPE_CONTROL with Depth Stall bit set, followed by a pipelined depth cache flush (PIPE_CONTROL with Depth Flush Bit set, followed by another pipelined depth stall (PIPE_CONTROL with Depth Stall Bit set), unless SW can otherwise guarantee that the pipeline from WM onwards is already flushed (e.g., via a preceding MI_FLUSH).

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: 07h 3DSTATE_HIER_DEPTH_BUFFER Format: OpCode
	15:8	**Reserved**
	7:0	**Dword Length** Format: =n Total Length - 2 Value: 3h — Name: Excludes Dword (0,1) **[Default]**

<table>
<tr><th colspan="3">3DSTATE_HIER_DEPTH_BUFFER</th></tr>
<tr><td>1</td><td>31:25</td><td>Hierarchical Depth Buffer Object Control State
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>MEMORY_OBJECT_CONTROL_STATE</td></tr></table>
Specifies the memory object control state for the hierarchical depth buffer.</td></tr>
<tr><td></td><td>24:23</td><td>Reserved
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>22:17</td><td>Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>16:0</td><td>Surface Pitch
<table><tr><td>Format:</td><td>U17-1 Pitch in Bytes</td></tr></table>
This field specifies the pitch of the hierarchical depth buffer in (#Bytes - 1).
<table><tr><th>Value</th><th>Name</th></tr><tr><td>[127, 1FFFFh]</td><td>corresponding to [128B, 128KB] also restricted to a multiple of 128B</td></tr></table>
<table><tr><th>Programming Notes</th></tr><tr><td>Since this surface is tiled, the pitch specified must be a multiple of the tile pitch, in the range [128B, 128KB].</td></tr></table></td></tr>
<tr><td>2..3</td><td>63:0</td><td>Surface Base Address
<table><tr><td>Format:</td><td>GraphicsAddress[63:0]HierarchicalDepthBuffer</td></tr></table>
This field specifies the address of the buffer in Graphics Memory.
<table><tr><th>Programming Notes</th></tr><tr><td>The Hierarchical Depth Buffer can only be mapped to Main Memory (uncached).</td></tr></table></td></tr>
<tr><td>4</td><td>31:15</td><td>Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>14:0</td><td>Surface QPitch
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>QPitch[16:2]</td></tr></table>
<table><tr><th>Description</th></tr><tr><td>This field specifies the distance in rows between array slices. It is used only in the following cases:

Surface Array is enabled OR
Number of Mulitsamples is not NUMSAMPLES_1 and Multisampled Surface Storage Format set to MSFMT_MSS OR
Surface Type is SURFTYPE_CUBE
</td></tr></table></td></tr>
</table>

3DSTATE_HIER_DEPTH_BUFFER

Value	Name	Description
[4h, 1FFFCh]		in multiples of 4 (low 2 bits missing)

Programming Notes
This field must be set to an integer multiple of 8 (QPitch[2] MBZ) Software must ensure that this field is set to a value sufficiently large such that the array slices in the surface do not overlap. Refer to the Memory Data Formats section for information on how surfaces are stored in memory.

3DSTATE_HS

3DSTATE_HS

Project: CHV, BSW
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** Project: All Format: MBZ
	7:0	**DWord Length** Format: =n Value: 7 — Name: Excludes DWord (0,1) **[Default]** — Project: CHV, BSW
1	31:30	**Reserved** Project: All Format: MBZ
	29:27	**Sampler Count** Project: All Format: U3 Specifies how many samplers (in multiples of 4) the HS kernels use. Used only for prefetching the associated sampler state entries.

3DSTATE_HS

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

26 **Reserved**

Project:	All
Format:	MBZ

25:18 **Binding Table Entry Count**

Project:	All
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	Project
When HW binding table bit is set, it is assumed that the Binding Table Entry Count field will be generated at JIT time.	CHV, BSW

17 **Thread Dispatch Priority**

Project:	CHV, BSW

Specifies the priority of the thread for dispatch

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

16 **Floating Point Mode**

Project:	All

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

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

15:14 **Reserved**

Project:	All
Format:	MBZ

3DSTATE_HS

	13	**Illegal Opcode Exception Enable** Project: All 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** Project: CHV, BSW Format: Enable This bit gets loaded into EU CRO1[13] (note the bit # difference). See Exceptions and ISA Execution Environment.
	11:8	**Reserved** Project: All Format: MBZ
	7:0	**Reserved** Project: CHV, BSW Format: MBZ
2	31	**Enable** Project: All 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** Project: All Format: Enable This bit controls whether HS-unit-specific statistics register(s) will increment (for each patch).
	28:18	**Reserved** Project: All Format: MBZ

3DSTATE_HS

DWord	Bit	Description
	17	**Reserved** Project: CHV, BSW Format: MBZ
	16:8	**Maximum Number of Threads** Project: CHV, BSW 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 | Project [0,503] | | indicating thread count of [1,504] | CHV, BSW [0,79] | | indicating thread count of [1,80] | CHV, BSW
	7:4	**Reserved** Format: MBZ
	3:0	**Instance Count** Format: U4-1 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** 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 **Project:** CHV, BSW	63:6	**Kernel Start Pointer** Project: CHV, BSW 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** Project: CHV, BSW Format: MBZ

3DSTATE_HS

DWord	Bit	Description
5..6 **Project:** CHV, BSW	63:10	**Scratch Space Base Pointer** Project: CHV, BSW 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** Project: CHV, BSW Format: MBZ
	3:0	**Per-Thread Scratch Space** Project: CHV, BSW 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 **Project:** CHV, BSW	31:29	**Reserved** Project: CHV, BSW Format: MBZ
	28	**Reserved** Format: MBZ
	27	**Single Program Flow** Project: CHV, BSW 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** Project: CHV, BSW Format: U1 Enumerated Type

3DSTATE_HS

		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**

Project:	CHV, BSW
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**

Project:	CHV, BSW
Format:	Boolean

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

Project:	CHV, BSW
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	Project
[0,31]		indicating GRF [R0, R31]	CHV, BSW

18:17 **Reserved**

Project:	CHV, BSW
Format:	MBZ

3DSTATE_HS

DWord	Bit	Description
	16:11	**Vertex URB Entry Read Length** Project: CHV, BSW Format: U6 Specifies the amount of URB data read and passed in the thread payload <u>for each Vertex URB entry</u>, 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** Project: CHV, BSW Format: MBZ
	9:4	**Vertex URB Entry Read Offset** Project: CHV, BSW 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** Project: CHV, BSW Format: MBZ
	0	**Reserved** Project: CHV, BSW Format: MBZ
8 **Project:** CHV, BSW	31:0	**Reserved** Project: CHV, BSW Format: MBZ

3DSTATE_INDEX_BUFFER

3DSTATE_INDEX_BUFFER

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command is used to specify the current IB state used by the VF function. At most one IB is defined and active at any given time.NOTES: The IB must be specified before any RANDOM 3D_PRIMITIVE commands are issued It is possible to have vertex elements source completely from generated ID values and therefore not require any Index Buffer accesses. In this case, VF function will simply ignore the Index Buffer state.

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: 0Ah 3DSTATE_INDEX_BUFFER Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 3h Excludes DWord (0,1) Project: All Format: =n Total Length - 2
1	31:10	**Reserved** Project: All Format: MBZ
	9:8	**Index Format** Project: All Format: U2 Enumerated type This field specifies the data format of the index buffer. All index values are UNSIGNED.

3DSTATE_INDEX_BUFFER

		<table><tr><th>Value</th><th>Name</th><th>Project</th></tr><tr><td>0h</td><td>BYTE</td><td>All</td></tr><tr><td>1h</td><td>WORD</td><td>All</td></tr><tr><td>2h</td><td>DWORD</td><td>All</td></tr></table>
	7	**Reserved** <table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table>
	6:0	**Memory Object Control State** <table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MEMORY_OBJECT_CONTROL_STATE</td></tr></table>Specifies the memory object control state for this index buffer.
2..3	63:0	**Buffer Starting Address** <table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>GraphicsAddress[63:0]Index_Buffer_Entry</td></tr></table>This field contains the size-aligned (as specified by Index Format) Graphics Address LSBs of the first element of interest within the index buffer. Software must program this value with the combination (sum) of the base address of the memory resource and the byte offset from the base address to the starting structure within the buffer. <table><tr><th>Programming Notes</th></tr><tr><td>Index Buffers can only be allocated in linear (not tiled) graphics memory.</td></tr></table>
4	31:0	**Buffer Size** <table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>U32 Count of bytes</td></tr></table>This field specifies the size of the buffer in bytes. Index accesses which straddle or go past the end of the buffer will return 0..Note that BufferSize=0 indicates that there is no valid data in the buffer. <table><tr><th>Value</th><th>Name</th></tr><tr><td>[0, FFFFFFFFh]</td><td></td></tr></table>

3DSTATE_LINE_STIPPLE

3DSTATE_LINE_STIPPLE		
Project:	CHV, BSW	
Source:	RenderCS	
Length Bias:	2	

The 3DSTATE_LINE_STIPPLE command is used to specify state variables used in the Line Stipple function.

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: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 08h 3DSTATE_LINE_STIPPLE Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**Dword Length** Default Value: 1h Excludes Dword (0,1) Project: All Format: =n Total Length - 2
1	31	**Modify Enable (Current Repeat Counter, Current Stipple Index)** Project: All Format: Enable Modify enable for **Current Repeat Counter** and **Current Stipple Index** fields. **Programming Notes** It is provided only for HW-generated commands as part of context save/restore. SW must initilize the current repeat counter, current stipple count fields if it sets this bit to enable. SW must set this bit to reset the stipple count.
	30	**Reserved**

<table>
<tr><th colspan="3">3DSTATE_LINE_STIPPLE</th></tr>
<tr><td></td><td>29:21</td><td>Current Repeat Counter<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>U9</td></tr></table>This field sets the HW-internal repeat counter state.
SW must initilize it to 1 if the modify enable is set.</td></tr>
<tr><td></td><td>20</td><td>Reserved<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>19:16</td><td>Current Stipple Index<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>U4</td></tr></table>This field sets the HW-internal stipple pattern index.
SW must initialize it to 0 if the modify enable is set.</td></tr>
<tr><td></td><td>15:0</td><td>Line Stipple Pattern<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>16 bit mask Bit 15 = most significant bit, Bit 0 = least significant bit</td></tr></table>Specifies a pattern used to mask out bit specific pixels while rendering lines.</td></tr>
<tr><td>2</td><td>31:15</td><td>Line Stipple Inverse Repeat Count<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>U1.16</td></tr></table><table><tr><td>Range: [0.00390625, 1.0]</td></tr><tr><td>Specifies the inverse (truncated) of the repeat count for the line stipple function.</td></tr></table></td></tr>
<tr><td></td><td>14:9</td><td>Reserved<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>8:0</td><td>Line Stipple Repeat Count<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>U9</td></tr></table>Specifies the repeat count for the line stipple function.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[1, 256]</td><td></td></tr></table></td></tr>
</table>

3DSTATE_MONOFILTER_SIZE

3DSTATE_MONOFILTER_SIZE

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This state specifies the size of the filter which is used when filtering in MAPFILTER_MONO mode.

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: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 11h 3DSTATE_MONOFILTER_SIZE Format: OpCode
	15:8	**Reserved**
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Project: All Format: =n Total Length - 2
1	31:6	**Reserved** Project: All Format: MBZ
	5:3	**Monochrome Filter Width** Project: All Format: U3 This field specifies the width of the monochrome filter. It is ignored if the monochrome filter is not enabled. Value: [1,7] — Name: (none)

3DSTATE_MONOFILTER_SIZE		
	2:0	**Monochrome Filter Height** Project: All Format: U3 This field specifies the height of the monochrome filter. It is ignored if the monochrome filter is not enabled.

Value	Name
[1,7]	

3DSTATE_MULTISAMPLE

3DSTATE_MULTISAMPLE

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The 3DSTATE_MULTISAMPLE command is used to specify multisample state associated with the current render target/depth buffer.

Programming Notes
It is illegal to render to surfaces with multiple different values of the state fields in this command.

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: 0Dh 3DSTATE_MULTISAMPLE Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**Dword Length** Default Value: 0h Project: All Format: =n Total Length - 2 Excludes Dword (0,1)
1	31:6	**Reserved** Project: All Format: MBZ

3DSTATE_MULTISAMPLE

	5	**Pixel Position Offset Enable**

Project:	CHV, BSW
Format:	U1

Enables the device to offset pixel positions by 0.5 both in horizontal and vertical directions.

Programming Notes
Setting this field along with setting the Pixel Location to upper left and number of multisamples to greater than one will cause the device to offset pixel postions by 0.5 both in horizontal and vertical directions. It is to be noted this is done to adjust the pixel co-ordinate system to DX9 like, so any WM_HZ_OP screen space rectangles (eg: legacy HiZ Clear, Resolve etc) generated internally by driver in this mode needs to be aware of this offset adjustment and send the rectangles according to alignment restriction taking this offset adjustment into consideration. SW can choose to set this bit only for DX9 API. DX10/OGL API's should not have any effect by setting or not setting this bit.

	4	**Pixel Location**

Project:	All
Format:	U1

This field specifies where the device evaluates "pixel" (vs. centroid or sample) values/attributes.

Value	Name	Description
0h	CENTER	Use the pixel center (0.5, 0.5 offset)
1h	UL_CORNER	Use the pixel upper-left corner

Programming Notes	Project
The programming of this field is assumed to be a function of the API being supported. Specifically, it is expected that OpenGL and DX10+ APIs require CENTER selection, while DX9- APIs require UL_CORNER selection.	
When 3DSTATE_RASTER::**ForcedSampleCount** is other than NUMRASTSAMPLES_0, this field must be 0h.	CHV, BSW

	3:1	**Number of Multisamples**

Project:	All
Format:	U3

This field specifies how many samples/pixel exist in all RTs and the Depth Buffer, as log2(#samples). This field is valid regardless of the setting of **Multisample Rasterization Mode**.

Value	Name	Description	Project
0h	1	1 sample/pixel	All
1h	2	2 samples/pixel	CHV, BSW
2h	4	4 samples/pixel	All
3h	8	8 samples/pixel	CHV, BSW
5h-7h	Reserved		All

3DSTATE_MULTISAMPLE		
		Programming Notes The setting of this field must match the **Number of Multisamples** field in SURFACE_STATE of all bound render targets.
	0	**Reserved** Project: All Format: MBZ

3DSTATE_POLY_STIPPLE_OFFSET

3DSTATE_POLY_STIPPLE_OFFSET

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The 3DSTATE_POLY_STIPPLE_OFFSET command is used to specify the origin of the repeated screen-space Polygon Stipple Pattern as an X, Y offset from the Color Buffer origin.

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: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 06h 3DSTATE_POLY_STIPPLE_OFFSET Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**Dword Length** Default Value: 0h Excludes Dword (0,1) Project: All Format: =n Total Length - 2
1	31:13	**Reserved** Project: All Format: MBZ
	12:8	**Polygon Stipple X Offset** Project: All Format: U5 Specifies a 5 bit x address offset in the poly stipple pattern Value: [0,31] / Name:

<table>
<tr><th colspan="3">3DSTATE_POLY_STIPPLE_OFFSET</th></tr>
<tr><td rowspan="2"></td><td>7:5</td><td>Reserved
<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>4:0</td><td>Polygon Stipple Y Offset
<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>U5</td></tr></table>
Specifies a 5 bit y address offset in the poly stipple pattern
<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,31]</td><td></td></tr></table></td></tr>
</table>

3DSTATE_POLY_STIPPLE_PATTERN

<table>
<tr><th colspan="3">3DSTATE_POLY_STIPPLE_PATTERN</th></tr>
<tr><td colspan="3">Project: CHV, BSW
Source: RenderCS
Length Bias: 2</td></tr>
<tr><td colspan="3">The 3DSTATE_POLY_STIPPLE_PATTERN command is used to specify the 32x32 Polygon Stipple Pattern used in the Polygon Stipple function of the WM unit.</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="6">0</td><td>31:29</td><td>Command Type<table><tr><td>Default Value:</td><td>3h GFXPIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td>28:27</td><td>Command SubType<table><tr><td>Default Value:</td><td>3h GFXPIPE_3D</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td>26:24</td><td>3D Command Opcode<table><tr><td>Default Value:</td><td>1h 3DSTATE_NONPIPELINED</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td>23:16</td><td>3D Command Sub Opcode<table><tr><td>Default Value:</td><td>07h 3DSTATE_POLY_STIPPLE_PATTERN</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td>15:8</td><td>Reserved<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>7:0</td><td>Dword Length<table><tr><td>Default Value:</td><td>1Fh Excludes Dword (0,1)</td></tr><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>=n Total Length - 2</td></tr></table></td></tr>
<tr><td>1..32</td><td>31:0</td><td>Pattern Row<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>32 bit mask Bit 31 = upper left corner, Bit 0 = upper right corner of first row.</td></tr></table>Specifies a pattern used by Polygon Stipple to mask out specific pixels of every 32x32 area rendered.</td></tr>
</table>

3DSTATE_PS_BLEND

3DSTATE_PS_BLEND

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

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: 4Dh 3DSTATE_PS_BLEND Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2 Value: 0h; Name: **[Default]**; Description: Excludes Dword (0,1)
1	31	**Alpha To Coverage Enable** Project: All Format: Enable If set, indicates that AlphaToCoverage is on RT[0], since this bit must be set the same for all RTs in the MRT case.
	30	**Has Writeable RT** Project: All Format: Enable When set indicates the there is at least one non-null RT w/ at least one channel write enabled

3DSTATE_PS_BLEND

	29	**Color Buffer Blend Enable** Project: All Format: Enable When set indicates that RT[0] has color buffer blend enabled.
	28:24	**Source Alpha Blend Factor** Project: All Format: 3D_Color_Buffer_Blend_Factor Indicates the "source factor" in alpha Color Buffer Blending stage for RT[0]
	23:19	**Destination Alpha Blend Factor** Project: All Format: 3D_Color_Buffer_Blend_Factor Indicates the "destination factor" in alpha Color Buffer Blending stage for RT[0]
	18:14	**Source Blend Factor** Project: All Format: 3D_Color_Buffer_Blend_Factor Indicates the "source factor" in Color Buffer Blending stage for RT[0]
	13:9	**Destination Blend Factor** Project: All Format: 3D_Color_Buffer_Blend_Factor Indicates the "destination factor" in Color Buffer Blending stage for RT[0]
	8	**Alpha Test Enable** Project: All Format: Enable Indicates the AlphaTestEnable for RT[0]
	7	**Independent Alpha Blend Enable** Project: All Format: Enable Indicates the Independent Alpha Blend Enable for RT[0] When enabled, the other fields in this instruction control the combination of the alpha components in the Color Buffer Blend stage. When disabled, the alpha components are combined in the same fashion as the color components.
	6:0	**Reserved**

3DSTATE_PS

3DSTATE_PS		
Project:	CHV, BSW	
Source:	RenderCS	
Length Bias:	2	

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: 20h 3DSTATE_PS Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0Ah Excludes DWord (0,1) Project: All Format: =n Total Length - 2
1..2	63:6	**Kernel Start Pointer 0** Project: All Format: InstructionBaseOffset[63:6]Kernel Specifies the 64-byte aligned address offset of the first instruction in the kernel[0]. This pointer is relative to the **Instruction Base Address.**
	5:0	**Reserved** Project: All Format: MBZ
3	31	**Single Program Flow** Project: All

3DSTATE_PS

		Single Program Flow (SPF) 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	Project
0h	Multiple	Multiple Program Flows	All
1h	Single	Single Program Flows	All

30 **Vector Mask Enable**

Project:	All
Format:	U1 Enumerated Type

When SPF=0, Vector Mask Enable (VME) specifies which mask to use to initialize the initial channel enables. When SPF=1, VME specifies which mask to use to generate execution channel enables.

Value	Name	Description	Project
0h	Dmask	Channels are enabled based on the dispatch mask	All
1h	Vmask	Channels are enabled based on the vector mask	All

29:27 **Sampler Count**

Project:	All

Specifies how many samplers (in multiples of 4) the vertex shader 0 kernel uses. Used only for prefetching the associated sampler state entries.

Value	Name	Description	Project
[0,4]			
0h	No Samplers	no samplers used	All
1h	1-4 Samplers	between 1 and 4 samplers used	All
2h	5-8 Samplers	between 5 and 8 samplers used	All
3h	9-12 Samplers	between 9 and 12 samplers used	All
4h	13-16 Samplers	between 13 and 16 samplers used	All
5h-7h		Reserved	All

26 **Single Precision Denormal Mode**

Project:	All

Specifies the single precision denornal mode used by the dispatched thread.

Value	Name	Description	Project
0h	Flushed to Zero	Single Precision denormals are flushed to zero	All
1h	Retained	Single Precision denormals are retained	All

25:18 **Binding Table Entry Count**

Project:	All

Description

3DSTATE_PS

		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 advantageous to set this field to zero to avoid prefetching too many entries and thrashing the state cache. This field is ignored if [PS Function Enable] is DISABLED. 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. See 3D Pipeline for more information. **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**
	16	**Floating Point Mode**
	15:14	**Rounding Mode**
	13	**Illegal Opcode Exception Enable**

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

Project:	All

Specifies the priority of the thread for dispatch.

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

16 — Floating Point Mode

Project:	All

Specifies the floating point mode used by the dispatched thread.

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

15:14 — Rounding Mode

Project:	All

Specifies the rounding mode used by the dispatched thread.

Value	Name	Description	Project
0h	RTNE	Round to Nearest Even	All
1h	RU	Round toward +infinity	All
2h	RD	Round toward -infinity	All
3h	RTZ	Round toward zero	All

13 — Illegal Opcode Exception Enable

Project:	All
Format:	Enable

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

3DSTATE_PS		
	12	**Reserved** Project: All Format: MBZ
	11	**Mask Stack Exception Enable** Project: All Format: Enable This bit gets loaded into EU CR0.1[12] (note the bit # difference). See Exceptions and ISA Execution Environment.
	10:8	**Reserved** Project: All Format: MBZ
	7	**Software Exception Enable** Project: All Format: Enable This bit gets loaded into EU CR0.1[13] (note the bit # difference). See Exceptions and ISA Execution Environment.
	6:0	**Reserved** Project: All Format: MBZ
4..5	63:10	**Scratch Space Base Pointer** Project: All Format: GeneralStateOffset[63:10]ScratchSpace Specifies the 1k-byte aligned address offset to scratch space for use by the kernel. This pointer is relative to the General State Base Address. **Programming Notes** Scratch Space per slice is computed based on 4 sub-slices. SW must allocate scratch space enough so that each slice has 4 slices allowed.
	9:4	**Reserved** Project: All Format: MBZ
	3:0	**Per Thread Scratch Space** Project: All Format: U4 Specifies the amount of scratch space allowed to be used by each thread. The driver must allocate enough contiguous scratch space, pointed to by the Scratch Space Pointer, to ensure that the Maximum Number of Threads each get Per Thread Scratch Space size without exceeding the driver-allocated scratch space.

3DSTATE_PS

DWord	Bit	Description
		<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,11]</td><td>indicating [1k bytes, 2M bytes] in powers of two</td></tr></table>
6	31:23	**Maximum Number of Threads Per PSD** <table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>U8-2 Representing Thread Count</td></tr></table><table><tr><th>Description</th></tr><tr><td>Range = [0, 62] --> [2, 64] threads. Specifies the maximum number of simultaneous threads allowed to be active per PSD. Depending on the GT mode register, Number of PSDs in the system determine maximum number PS threads per configuration.</td></tr><tr><td>Has 2 PSDs therefore actual range for max PS threads is [4,128].</td></tr></table><table><tr><th>Programming Notes</th></tr><tr><td>If this field is changed between 3DPRIMITIVE commands, a PIPE_CONTROL command with Stall at Pixel Scoreboard set is required to be issued.</td></tr></table>
	22:12	**Reserved** <table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table>
	11	**Push Constant Enable** <table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>Enable</td></tr></table>This field must be enabled if the sum of the **PS Constant Buffer [3:0] Read Length** fields in 3DSTATE_CONSTANT_PS is nonzero, and must be disabled if the sum is zero.
	10	**Reserved** <table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table>
	9	**Reserved** <table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table>
	8	**Render Target Fast Clear Enable** <table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>Enable</td></tr></table>This field is set to enable fast clear of the bound render targets. See "Render Target Fast Clear" for restrictions on enabling this field.
	7	**Reserved** <table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table>

3DSTATE_PS

	6	**Render Target Resolve Enable** Project: CHV, BSW Format: Enable This field is set to enable clear value resolve on non-multisampled render targets. See "Render Target Resolve" for restrictions on enabling this field.
	5	**Reserved** Project: Format: MBZ
	4:3	**Position XY Offset Select** Project: All Format: U2 Enumerated Type This field specifies if/what Position XY Offset values are passed in the PS payload. Note that these are per-slot (pixel|sample) offsets, and therefore separate from the subspan XY coordinates passed in R1.

Value	Name	Description
0h	POSOFFSET_NONE	No Position XY Offsets are included in the PS payload.
1h	Reserved	
2h	POSOFFSET_CENTROID	Position XY Offsets will be passed in the PS payload, and these will reflect the Centroid position(s).
3h	POSOFFSET_SAMPLE	Position XY Offsets will be passed in the PS payload, and these will reflect the multisample position(s).

Programming Notes
SW Recommendation: If the PS kernel needs the Position Offsets to compute a Position XY value, this field should match Position ZW Interpolation Mode to ensure a consistent position.xyzw computation
If the PS kernel does not need the Position XY Offsets to compute a Position Value, then this field should be programmed to POSOFFSET_NONE, as the PS kernel should be using the various barycentric inputs to evaluate other-than-position attributes. However, this field can be used to pass Centroid or Sample offsets in the payload for special test modes (e.g., where barycentric coordinates are computed in the PS vs. being HW-generated and passed in the payload).
MSDISPMODE_PERSAMPLE is required in order to select POSOFFSET_SAMPLE.

	2	**32 Pixel Dispatch Enable** Project: All Format: Enable Enables the Windower to dispatch 8 subspans in one payload. Variable Pixel Dispatch in Section: Pixel Grouping (Dispatch size) control for valid pixel dispatch combinations.

3DSTATE_PS

DWord	Bit	Description
	1	**16 Pixel Dispatch Enable** Project: All Format: Enable Enables the Windower to dispatch 4 subspans in one payload. Variable Pixel Dispatch in Section: Pixel Grouping (Dispatch size) control for valid pixel dispatch combinations.
	0	**8 Pixel Dispatch Enable** Project: All Format: Enable Enables the Windower to dispatch 2 subspans from 1 object (polygon) in one payload. Variable Pixel Dispatch in Section: Pixel Grouping (Dispatch size) control for valid pixel dispatch combinations. **Programming Notes** / **Project** When Render Target Fast Clear Enable is ENABLED or Render Target Resolve Type = RESOLVE_PARTIAL or RESOLVE_FULL, this bit must be DISABLED. / CHV, BSW
7	31:23	**Reserved** Project: All Format: MBZ
	22:16	**Dispatch GRF Start Register For Constant/Setup Data 0** Format: U7 Specifies the starting GRF register number for the Constant/Setup portion of the thread payload for kernel[0]. **Value** / **Name** [0,127] /
	15	**Reserved** Project: All Format: MBZ
	14:8	**Dispatch GRF Start Register For Constant/Setup Data 1** Project: All Format: U7 Specifies the starting GRF register number for the Constant/Setup portion of the thread payload for kernel[1]. **Value** / **Name** [0,127] /
	7	**Reserved** Project: All Format: MBZ

3DSTATE_PS

	6:0	**Dispatch GRF Start Register For Constant/Setup Data 2** Project: All Format: U7 Specifies the starting GRF register number for the Constant/Setup portion of the thread payload for kernel[2]. **Value** / **Name** [0,127] /
8..9	63:6	**Kernel Start Pointer 1** Project: All Format: InstructionBaseOffset[63:6]Kernel Specifies the 64-byte aligned address offset of the first instruction in kernel[1]. This pointer is relative to the Instruction Base Address.
	5:0	**Reserved** Project: All Format: MBZ
10..11	63:6	**Kernel Start Pointer 2** Project: All Format: InstructionBaseOffset[63:6]Kernel Specifies the 64-byte aligned address offset of the first instruction in kernel[2]. This pointer is relative to the **Instruction Base Address**.
	5:0	**Reserved** Project: All Format: MBZ

3DSTATE_PS_EXTRA

3DSTATE_PS_EXTRA

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

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: 4fh 3DSTATE_PS_EXTRA Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Project: All Format: =n Total Length - 2
1	31	**Pixel Shader Valid** Project: All Format: Enable When set indicates a valid pixel shaderWhen this bit clear the rest of this command should also be clear.
	30	**Pixel Shader Does not write to RT** Project: All Format: Enable When set indicates the pixel shader does not write to render target.

3DSTATE_PS_EXTRA

	29	**oMask Present to Render Target** Project: All Format: Enable This bit is inserted in the PS payload header and made available to the DataPort (either via the message header or via header bypass) to indicate that oMask data from the shader (one or two phases) is included in Render Target Write messages. If present, the oMask data is used to mask off samples.
	28	**Pixel Shader Kills Pixel** Project: All Format: Enable This bit, if ENABLED, indicates that the PS kernel has the ability to kill (discard) pixels or samples, other than due to depth or stencil testing. This bit is required to be ENABLED in the following situations: • The API pixel shader program contains "killpix" or "discard" instructions, or other code in the pixel shader kernel that can cause the final pixel mask to differ from the pixel mask received on dispatch.
	27:26	**Pixel Shader Computed Depth Mode** Project: All Format: U2 Enumerated Type This field specifies the computed depth mode for the pixel shader. (see value table below)
	25	**Force Computed Depth** Project: All Format: Enable (see Programming Notes below)

Pixel Shader Computed Depth Mode values:

Value	Name	Description
0h	PSCDEPTH_OFF	Pixel shader does not compute depth
1h	PSCDEPTH_ON	Pixel shader computes depth with no guarantee as to its value
2h	PSCDEPTH_ON_GE	Pixel shader computes depth and guarantees that oDepth >= SourceDepth
3h	PSCDEPTH_ON_LE	Pixel shader computes depth and guarantees that oDepth <= SourceDepth

Programming Notes
If this field is set to any value other than PSCDEPTH_OFF, a multi-phase shader (i.e. dispatch RATE_COARSE or RATE_PIXEL with pixel/sample loops or sample loop respectively) must output depth and render targets only at the last phase.

Force Computed Depth:

Programming Notes
This field should be left DISABLED, except for driver debug. This field should not be tested for functional validation.

3DSTATE_PS_EXTRA

	24	**Pixel Shader Uses Source Depth** Project: All Format: Enable This bit, if ENABLED, indicates that the PS kernel requires the source depth value (vPos.z) to be passed in the payload. The source depth value is interpolated according to the Position ZW Interpolation Mode state.
	23	**Pixel Shader Uses Source W** Project: All Format: Enable This bit, if ENABLED, indicates that the PS kernel requires the interpolated source W value (vPos.w) to be passed in the payload. The W value is interpolated according to the Position ZW Interpolation Mode state.
	22	**Reserved** Project: CHV, BSW Format: MBZ
	21:18	**Reserved** Project: CHV, BSW Format: MBZ
	17	**Removed** Project: CHV, BSW Format: MBZ
	16:11	**Reserved**
	10	**Reserved** Project: CHV, BSW Format: MBZ
	9	**Reserved** Project: CHV, BSW
	8	**Attribute Enable** Format: Enable This field must be enabled if the Number of SF Output Attributes field in 3DSTATE_SBE is nonzero, and must be disabled if that field is zero.
	7	**Pixel Shader Disables Alpha To Coverage** Project: All Format: Enable When set indicates the pixel shader AlphaToCoverage should be disabled due to oMask output. The setting of this bit is API dependent.

3DSTATE_PS_EXTRA

	6	**Pixel Shader Is Per Sample** Project: CHV, BSW Format: Enable This bit, when ENABLED, indicates that the pixel shader is dispatched at the per sample shading rate. If the bit is DISABLED, the pixel shader is dispatched at the per pixel rate. **Programming Notes** This bit must NOT be set when PS is used to do clear MSRTs with Fast Clear Optimization Enabled.
	5	**Reserved** Project: CHV, BSW Format: MBZ
	4	**Reserved** Project: CHV, BSW Format: MBZ
	3	**Reserved** Project: CHV, BSW Format: MBZ
	2	**Pixel Shader Has UAV** Project: All Format: Enable Format: U1 Enumerated Type This field when set indicates that the pixel shader has a UAV attached to it.
	1	**Pixel Shader Uses Input Coverage Mask** Project: CHV, BSW Format: Enable This bit, if ENABLED, indicates that the PS kernel requires the input coverage mask to be passed in the payload.
	0	**Reserved** Project: CHV, BSW Format: MBZ

3DSTATE_PUSH_CONSTANT_ALLOC_DS

3DSTATE_PUSH_CONSTANT_ALLOC_DS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command sets up the URB configuration for DS Push Constant Buffer.

Programming Notes

Programming Restriction:

- The sum of the Constant Buffer Offset and the Constant Buffer Size may not exceed the maximum value of the Constant Buffer Size.
- The sum of the constant length programmed in 3DSTATE_CONSTANT_DS must be equal or smaller then the size of the allocated space in the URB including the buffering for half cachelines. See **Push Constant URB Allocation section for more details**.
- The 3DSTATE_CONSTANT_DS must be reprogrammed prior to the next 3DPRIMITIVE command after programming the 3DSTATE_PUSH_CONSTANT_ALLOC_DS.

DWord	Bit	Description
0 **Programming Notes:** Workaround: This command must be followed by a PIPE_CONTROL with CS Stall bit set.	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: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 14h 3DSTATE_PUSH_CONSTANT_ALLOC_DS Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Project: All Format: =n Total Length - 2

<table>
<tr><th colspan="3">3DSTATE_PUSH_CONSTANT_ALLOC_DS</th></tr>
<tr><td rowspan="4">1</td><td>31:21</td><td>Reserved
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>20:16</td><td>Constant Buffer Offset
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>U5</td></tr></table>Specifies the offset of the DS constant buffer into the URB.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,31]</td><td>(0KB - 31KB) Increments of 2KB</td></tr></table></td></tr>
<tr><td>15:6</td><td>Reserved
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>5:0</td><td>Constant Buffer Size
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>U6</td></tr></table>Specifies the size of the DS constant buffer. This value will determine the amount of data the command stream can pre-fetch before the buffer is full. Value of zero is only valid when constants are not enabled for DS.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,32]</td><td>(0KB - 32KB) Increments of 2KB</td></tr></table></td></tr>
</table>

3DSTATE_PUSH_CONSTANT_ALLOC_GS

3DSTATE_PUSH_CONSTANT_ALLOC_GS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command sets up the URB configuration for GS Push Constant Buffer.

Programming Notes

- The sum of the Constant Buffer Offset and the Constant Buffer Size may not exceed the maximum value of the Constant Buffer Size.
- The sum of the constant length programmed in 3DSTATE_CONSTANT_GS must be equal or smaller then the size of the allocated space in the URB including the buffering for half cachelines.
- The 3DSTATE_CONSTANT_GS must be reprogrammed prior to the next 3DPRIMITIVE command after programming the 3DSTATE_PUSH_CONSTANT_ALLOC_GS.

See Push Constant URB Allocation section for more details.

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: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 15h 3DSTATE_PUSH_CONSTANT_ALLOC_GS Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2

Value	Name	Description
0h	3DSTATE_PUSH_CONSTANT_ALLOC_GS **[Default]**	Excludes DWord (0,1)

3DSTATE_PUSH_CONSTANT_ALLOC_GS		
1	31:21	**Reserved** Project: CHV, BSW Format: MBZ
	20:16	**Constant Buffer Offset** Project: CHV, BSW Format: U5 Specifies the offset of the GS constant buffer into the URB. Value: [0,31] — Name: (0KB - 31KB) Increments of 2KB — Project: CHV, BSW
	15:6	**Reserved** Project: CHV, BSW Format: MBZ
	5:0	**Constant Buffer Size** Project: CHV, BSW Format: U6 Specifies the size of the GS constant buffer. This value will determine the amount of data the command stream can pre-fetch before the buffer is full. Value of zero is only valid when constants are not enabled for GS. Value: [0,32] — Name: (0KB - 32KB) Increments of 2KB — Project: CHV, BSW

3DSTATE_PUSH_CONSTANT_ALLOC_HS

3DSTATE_PUSH_CONSTANT_ALLOC_HS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command sets up the URB configuration for HS Push Constant Buffer.

Programming Notes

Programming Restriction:

- The sum of the Constant Buffer Offset and the Constant Buffer Size may not exceed the maximum value of the Constant Buffer Size.
- The sum of the constant length programmed in 3DSTATE_CONSTANT_HS must be equal or smaller then the size of the allocated space in the URB including the buffering for half cachelines. See **Push Constant URB Allocation section for more details**.
- The 3DSTATE_CONSTANT_HS must be reprogrammed prior to the next 3DPRIMITIVE command after programming the 3DSTATE_PUSH_CONSTANT_ALLOC_HS.

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:	1h 3DSTATE_NONPIPELINED
		Format:	OpCode
	23:16	**3D Command Sub Opcode**	
		Default Value:	13h 3DSTATE_PUSH_CONSTANT_ALLOC_HS
		Format:	OpCode
	15:8	**Reserved**	
		Project:	All
		Format:	MBZ
	7:0	**DWord Length**	
		Default Value:	0h Excludes DWord (0,1)
		Project:	All
		Format:	=n Total Length - 2

3DSTATE_PUSH_CONSTANT_ALLOC_HS

<table>
<tr><td>1</td><td>31:21</td><td>Reserved
Project: CHV, BSW
Format: MBZ</td></tr>
<tr><td></td><td>20:16</td><td>Constant Buffer Offset
Project: CHV, BSW
Format: U5
Specifies the offset of the HS constant buffer into the URB.
<table><tr><th>Value</th><th>Name</th><th>Project</th></tr><tr><td>[0,31]</td><td>(0KB - 31KB) Increments of 2KB</td><td>CHV, BSW</td></tr></table></td></tr>
<tr><td></td><td>15:6</td><td>Reserved
Project: CHV, BSW
Format: MBZ</td></tr>
<tr><td></td><td>5:0</td><td>Constant Buffer Size
Project: CHV, BSW
Format: U6
Specifies the size of the HS constant buffer. This value will determine the amount of data the command stream can pre-fetch before the buffer is full. Value of zero is only valid when constants are not enabled for HS.
<table><tr><th>Value</th><th>Name</th><th>Project</th></tr><tr><td>[0,32]</td><td>(0KB - 32KB) Increments of 2KB</td><td>CHV, BSW</td></tr></table></td></tr>
</table>

3DSTATE_PUSH_CONSTANT_ALLOC_PS

3DSTATE_PUSH_CONSTANT_ALLOC_PS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command sets up the URB configuration for PS Push Constant Buffer.

Programming Notes

Restriction:

- The sum of the Constant Buffer Offset and the Constant Buffer Size may not exceed the maximum value of the Constant Buffer Size.
- The sum of the constant length programmed in 3DSTATE_CONSTANT_PS must be equal or smaller then the size of the allocated space in the URB including the buffering for half cachelines. See **Push Constant URB Allocation** section for more details.
- The 3DSTATE_CONSTANT_PS must be reprogrammed prior to the next 3DPRIMITIVE command after programming the 3DSTATE_PUSH_CONSTANT_ALLOC_PS.

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:	1h 3DSTATE_NONPIPELINED
		Format:	OpCode
	23:16	**3D Command Sub Opcode**	
		Default Value:	16h 3DSTATE_PUSH_CONSTANT_ALLOC_PS
		Format:	OpCode
	15:8	**Reserved**	
		Project:	All
		Format:	MBZ
	7:0	**Dword Length**	
		Default Value:	0h Excludes Dword (0,1)
		Project:	All
		Format:	=n Total Length - 2

<table>
<tr><th colspan="3">3DSTATE_PUSH_CONSTANT_ALLOC_PS</th></tr>
<tr><td rowspan="4">1</td><td>31:21</td><td>Reserved
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>20:16</td><td>Constant Buffer Offset
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>U5</td></tr></table>
Specifies the offset of the PS constant buffer into the URB.
<table><tr><th>Value</th><th>Name</th><th>Project</th></tr><tr><td>[0,31]</td><td>(0KB - 31KB) Increments of 2KB</td><td>CHV, BSW</td></tr></table></td></tr>
<tr><td>15:6</td><td>Reserved
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>5:0</td><td>Constant Buffer Size
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>U6</td></tr></table>
Specifies the size of the PS constant buffer. This value will determine the amount of data the command stream can pre-fetch before the buffer is full. Value of zero is only valid when constants are not enabled for PS.
<table><tr><th>Value</th><th>Name</th><th>Project</th></tr><tr><td>[0,32]</td><td>(0KB - 32KB) Increments of 2KB</td><td>CHV, BSW</td></tr></table></td></tr>
</table>

3DSTATE_PUSH_CONSTANT_ALLOC_VS

3DSTATE_PUSH_CONSTANT_ALLOC_VS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command sets up the URB configuration for VS Push Constant Buffer.

Programming Notes

Programming Restriction:

- The sum of the Constant Buffer Offset and the Constant Buffer Size may not exceed the maximum value of the Constant Buffer Size.
- The sum of the constant length programmed in 3DSTATE_CONSTANT_VS must be equal or smaller then the size of the allocated space in the URB including the buffering for half cachelines. See **Push Constant URB Allocation section for more details**.
- The 3DSTATE_CONSTANT_VS must be reprogrammed prior to the next 3DPRIMITIVE command after programming the 3DSTATE_PUSH_CONSTANT_ALLOC_VS.

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: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 12h 3DSTATE_PUSH_CONSTANT_ALLOC_VS Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Project: All Format: =n Total Length - 2

3DSTATE_PUSH_CONSTANT_ALLOC_VS

1	31:21	**Reserved** Project: CHV, BSW Format: MBZ
	20:16	**Constant Buffer Offset** Project: CHV, BSW Format: U5 Specifies the offset of the VS constant buffer into the URB. Value: [0,31] — Name: (0KB - 31KB) Increments of 2KB — Project: CHV, BSW
	15:6	**Reserved** Project: CHV, BSW Format: MBZ
	5:0	**Constant Buffer Size** Project: CHV, BSW Format: U6 Specifies the size of the VS constant buffer. This value will determine the amount of data the command stream can pre-fetch before the buffer is full. Value of zero is only valid when constants are not enabled for VS. Value: [0,32] — Name: (0KB - 32KB) Increments of 2KB — Project: CHV, BSW

3DSTATE_RASTER

3DSTATE_RASTER

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

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:	50h 3DSTATE_RASTER
		Format:	OpCode
	15:8	**Reserved**	
		Project:	All
		Format:	MBZ
	7:0	**DWord Length**	
		Default Value:	03h Excludes DWord (0,1)
		Project:	All
		Format:	=n
		Total Length - 2	
1	31:28	**Reserved**	
		Project:	All
		Format:	MBZ
	27	**Reserved**	
		Project:	CHV, BSW
		Format:	MBZ
	26:24	**Reserved**	
		Project:	CHV, BSW
		Format:	MBZ

3DSTATE_RASTER

	23:22	**API Mode** Project: All Software sets this field according to the API's version. These bits are set for DX9 or OGL/DX10.0/DX10.1+/DX11.1 per the following values.

Value	Name	Project
0h	DX9/OGL	All
1h	DX10.0	All
2h	DX10.1+	All
3h	Reserved	All

	21	**Front Winding** Project: All Determines whether a triangle object is considered "front facing" if the screen space vertex positions, when traversed in the order, result in a clockwise (CW) or counter-clockwise (CCW) winding order. Does not apply to points or lines.

Value	Name	Description	Project
0h	Clockwise	FRONTWINDING_CW	All
1h	Counter Clockwise **[Default]**	FRONTWINDING_CCW	All

	20:18	**Forced Sample Count** Project: All Format: U3 Enumerated Type This field specifies how many samples/pixel exist for RT Independent Rasterization

Value	Name	Description	Project
0h	NUMRASTSAMPLES_0	No RT Independent Rasterization	All
1h	NUMRASTSAMPLES_1	1 rast-sample/pixel	All
2h	NUMRASTSAMPLES_2	2 rast-samples/pixel	All
3h	NUMRASTSAMPLES_4	4 rast-samples/pixel	All
4h	NUMRASTSAMPLES_8	8 rast-samples/pixel	All
5h	NUMRASTSAMPLES_16	16 rast-samples/pixel	All
6h-7h	Reserved		All

Programming Notes
When 3DSTATE_MULTISAMPLE::Number of Multisamples !=NUMSAMPLES_1, this field mus be either NUMRASTSAMPLES_0 or NUMRASTSAMPLES_1. When 3DSTATE_MULTISAMPLE::Number of Multisamples == NUMSAMPLES_1, this field mus not be NUMRASTSAMPLES_1.

	17:16	**Cull Mode** Project: All Format: 3D_CullMode

3DSTATE_RASTER

		Controls removal (culling) of triangle objects based on orientation. The cull mode only applies to triangle objects and does not apply to lines, points or rectangles.

Value	Name	Description	Project
0h	CULLMODE_BOTH	All triangles are discarded (i.e., no triangle objects are drawn)	All
1h	CULLMODE_NONE **[Default]**	No triangles are discarded due to orientation	All
2h	CULLMODE_FRONT	Triangles with a front-facing orientation are discarded	All
3h	CULLMODE_BACK	Triangles with a back-facing orientation are discarded	All

Programming Notes
Orientation determination is based on the setting of the Front Winding state.

15 **Reserved**

Project:	All
Format:	MBZ

14 **Force Multisampling**

Project:	All

This field provides a work around override for the computation of SF_INT::Multisample Rasterization Mode and WM_INT::Multisample Rasterization Mode.

Value	Name	Description	Project
0h	Normal	Multisampling mode is computed by HW according to formula for signal SF_INT::Multisample Rasterization Mode and WM_INT::Multisample Rasterization Mode in vol2a.11 3D Pipeline Windower [CHV, BSW] > Windower Pipelined State > 3DSTATE_WM > 3DSTATE_WM [CHV, BSW].	All
1h	Force	Forces the DX Multisampling mode to be used directly	All

13 **Smooth Point Enable**

Project:	CHV, BSW
Format:	Enable

Software sets this according to API. When OGL and smooth point rasterization is required, this bit must be set. HW ignores this bit for primitives other than points.

12 **DX Multisample Rasterization Enable**

Project:	All
Format:	Enable

Software sets this according to the API's multisample enable

Programming Notes

3DSTATE_RASTER

		This state only effects how the SF_INT/WM_INT::Multisample Rasterization Mode are set depending on some other states. This state mainly modifies the how the line rendering is done by setting SF_INT/WM_INT::Multisample Rasterization Mode to either OFF* or ON* . Please refer to table under SF_INT::Multisample Rasterization Mode.
	11:10	**DX Multisample Rasterization Mode** Project: All Format: U2 enumerated type This field determines whether multisample rasterization is turned on/off, and how the pixel sample point(s) are defined. Software sets this according to the API's multisample state setting (if any)

Value	Name	Project
0h	MSRASTMODE_ OFF_PIXEL	All
1h	MSRASTMODE_ OFF_PATTERN	All
2h	MSRASTMODE_ ON_PIXEL	All
3h	MSRASTMODE_ ON_PATTERN	All

Programming Notes
This field is used to directly set the SF_INT/WM_INT::Multisample Rasterization Mode when DX Multisample Rasterization Enable is set. Please refer to equation of SF_INT::Multisample Rasterization Mode.

	9	**Global Depth Offset Enable Solid** Project: All Format: Enable Enables computation and application of Global Depth Offset for SOLID objects.
	8	**Global Depth Offset Enable Wireframe** Project: All Format: Enable Enables computation and application of Global Depth Offset when triangles are rendered in WIREFRAME mode.
	7	**Global Depth Offset Enable Point** Project: All Format: Enable Enables computation and application of Global Depth Offset when triangles are rendered in POINT mode.
	6:5	**Front Face Fill Mode** Project: All

3DSTATE_RASTER

		Format: U2 enumerated type This state controls how front-facing triangle and rectangle objects are rendered.

Value	Name	Description	Project
0h	SOLID	Any triangle or rectangle object found to be front-facing is rendered as a solid object. This setting is required when rendering rectangle (RECTLIST) objects.	All
1h	WIREFRAME	Any triangle object found to be front-facing is rendered as a series of lines along the triangle boundaries (as determined by the topology type and controlled by the vertex EdgeFlags).	All
2h	POINT	Any triangle object found to be front-facing is rendered as a set of point primitives at the triangle vertices (as determined by the topology type and controlled by the vertex EdgeFlags).	All
3h	Reserved		All

4:3 Back Face Fill Mode

Project:	All
Format:	U2 enumerated type

This state controls how back-facing triangle and rectangle objects are rendered.

Value	Name	Description	Project
0h	SOLID	Any triangle or rectangle object found to be back-facing is rendered as a solid object. This setting is required when rendering rectangle (RECTLIST) objects.	All
1h	WIREFRAME	Any triangle object found to be back-facing is rendered as a series of lines along the triangle boundaries (as determined by the topology type and controlled by the vertex EdgeFlags).	All
2h	POINT	Any triangle object found to be back-facing is rendered as a set of point primitives at the triangle vertices (as determined by the topology type and controlled by the vertex EdgeFlags).	All
3h	Reserved		All

2 Antialiasing Enable

Project:	All
Format:	Enable

This field enables "alpha-based" line antialiasing.

Programming Notes
This field must be disabled if any of the render targets have integer (UINT or SINT) surface format.

1 Scissor Rectangle Enable

Project:	All
Format:	Enable

Enables operation of Scissor Rectangle.

3DSTATE_RASTER		
	0	**Viewport Z Clip Test Enable** Project: CHV, BSW Format: Enable This field is used to control whether the Viewport Z extents (near, far) are considered in VertexClipTest.
2	31:0	**Global Depth Offset Constant** Format: IEEE_Float Specifies the constant term in the Global Depth Offset function.
3	31:0	**Global Depth Offset Scale** Format: IEEE_Float Specifies the scale term used in the Global Depth Offset function.
4	31:0	**Global Depth Offset Clamp** Format: IEEE_Float Specifies the clamp term used in the Global Depth Offset function.

3DSTATE_SAMPLE_MASK

3DSTATE_SAMPLE_MASK

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

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: 18h 3DSTATE_DX9_LOCAL_VALID_PS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**Dword Length** Default Value: 0h Excludes Dword (0,1) Format: =n Total Length - 2
1	31:16	**Reserved** Project: CHV, BSW Format: MBZ
	15:0	**Sample Mask** Project: CHV, BSW Format: 16 bit mask Right-justified bitmask (Bit 0 = Sample0). Number of bits that are used is determined by Num Multisamples (3DSTATE_MULTISAMPLE) A per-multisample-position mask state variable that is immediately and unconditionally ANDed with the sample coverage mask as part of the rasterization process. This mask is applied prior to centroid selection. This mask must be ignored for centroid selection when RTIR is enabled i.e. Forced_Sample_Count > 0.

3DSTATE_SAMPLE_MASK		
		Programming Notes • If **Number of Multisamples** is NUMSAMPLES_1, bits 15:1 of this field will be zeroed by HW. • If N**Number of Multisamples** is NUMSAMPLES_2, bits 15:2 of this field will be zeroed by HW. • If **Number of Multisamples** is NUMSAMPLES_4, bits 15:4 of this field will be zeroed by HW. • If **Number of Multisamples** is NUMSAMPLES_8, bits 15:8 of this field will be zeroed by HW.

3DSTATE_SAMPLE_PATTERN

3DSTATE_SAMPLE_PATTERN

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The 3DSTATE_SAMPLE_PATTERN command is used to specify the sample offsets for all multisample sample modes. The set of offset used will be selected based on the multisample mode. This is non-pipelined state.

Programming Notes
When programming the sample offsets (for NUMSAMPLES_4 or _8 and MSRASTMODE_xxx_PATTERN), the order of the samples 0 to 3 (or 7 for 8X, or 15 for 16X) must have monotonically increasing distance from the pixel center. This is required to get the correct centroid computation in the device.

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: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 1Ch 3DSTATE_SAMPLE_PATTERN Format: OpCode
	15:8	**Reserved**
	7:0	**Dword Length** Default Value: 7 Format: =n Total Length - 2 Excludes Dword (0,1)
1..4	31:0	**Reserved**
5	31:28	**8x Sample7 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 7 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]

3DSTATE_SAMPLE_PATTERN

	27:24	**8x Sample7 Y Offset** Project: All Format: U0.4 Subpixel Y offset of Sample 7 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	23:20	**8x Sample6 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 6 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	19:16	**8x Sample6 Y Offset** Project: All Format: U0.4 Subpixel Y offset of Sample 6 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	15:12	**8x Sample5 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 5 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	11:8	**8x Sample5 Y Offset** Project: All Format: U0.4 Subpixel Y offset of Sample 5 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	7:4	**8x Sample4 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 4 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]

3DSTATE_SAMPLE_PATTERN		
	3:0	**8x Sample4 Y Offset** Project: All Format: U0.4 Subpixel Y offset of Sample 4 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
6	31:28	**8x Sample3 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 3 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	27:24	**8x Sample3 Y Offset** Project: All Format: U0.4 Subpixel Y offset of Sample 3 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	23:20	**8x Sample2 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 2 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	19:16	**8x Sample2 Y Offset** Project: All Format: U0.4 Subpixel Y offset of Sample 2 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	15:12	**8x Sample1 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 1 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]

3DSTATE_SAMPLE_PATTERN		
	11:8	**8x Sample1 Y Offset** Project: All Format: U0.4 Subpixel Y offset of Sample 1 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	7:4	**8x Sample0 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 0 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	3:0	**8x Sample0 Y Offset** Project: All Format: U0.4 Subpixel Y offset of Sample 0 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
7	31:28	**4x Sample3 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 3 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]
	27:24	**4x Sample3 Y Offset** Project: All Format: U0.4 Subpixel Y offset of Sample 3 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]
	23:20	**4x Sample2 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 2 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]

3DSTATE_SAMPLE_PATTERN		
	19:16	**4x Sample2 Y Offset** Project: All Format: U0.4 Subpixel Y offset of Sample 2 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]
	15:12	**4x Sample1 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 1 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]
	11:8	**4x Sample1 Y Offset** Project: All Format: U0.4 Subpixel Y offset of Sample 1 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]
	7:4	**4x Sample0 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 0 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]
	3:0	**4x Sample0 Y Offset** Project: All Format: U0.4 Subpixel Y offset of Sample 0 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]
8	31:24	**Reserved**
	23:20	**1x Sample0 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 0 relative to the UL pixel origin for 1x mode. Range: [0,0.9375]

3DSTATE_SAMPLE_PATTERN		
	19:16	**1x Sample0 Y Offset** Project: All Format: U0.4 Subpixel Y offset of Sample 0 relative to the UL pixel origin for 1x mode. Range: [0,0.9375]
	15:12	**2x Sample1 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 1 relative to the UL pixel origin for 2x mode. Range: [0,0.9375]
	11:8	**2x Sample1 Y Offset** Project: All Format: U0.4 Subpixel Y offset of Sample 1 relative to the UL pixel origin for 2x mode. Range: [0,0.9375]
	7:4	**2x Sample0 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 0 relative to the UL pixel origin for 2x mode. Range: [0,0.9375]
	3:0	**2x Sample0 Y Offset** Project: All Format: U0.4 Subpixel Y offset of Sample 0 relative to the UL pixel origin for 2x mode. Range: [0,0.9375]

3DSTATE_SAMPLE_PATTERN

3DSTATE_SAMPLE_PATTERN

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The 3DSTATE_SAMPLE_PATTERN command is used to specify the sample offsets for all multisample sample modes. The set of offset used will be selected based on the multisample mode. This is non-pipelined state.

Programming Notes
When programming the sample offsets (for NUMSAMPLES_4 or _8 and MSRASTMODE_xxx_PATTERN), the order of the samples 0 to 3 (or 7 for 8X) must have monotonically increasing distance from the pixel center. This is required to get the correct centroid computation in the device.

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: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 1Ch 3DSTATE_SAMPLE_PATTERN Format: OpCode
	15:8	**Reserved**
	7:0	**Dword Length** Default Value: 3 Project: All Format: =n Total Length - 2 Excludes Dword (0,1)
1	31:28	**8x Sample7 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 7 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]

3DSTATE_SAMPLE_PATTERN		
	27:24	**8x Sample7 Y Offset** Project: All Format: U0.4 Subpixel Y offset of Sample 7 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	23:20	**8x Sample6 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 6 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	19:16	**8x Sample6 Y Offset** Project: All Format: U0.4 Subpixel Y offset of Sample 6 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	15:12	**8x Sample5 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 5 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	11:8	**8x Sample5 Y Offset** Format: U0.4 Subpixel Y offset of Sample 5 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	7:4	**8x Sample4 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 4 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	3:0	**8x Sample4 Y Offset** Project: All Format: U0.4

3DSTATE_SAMPLE_PATTERN

		Subpixel Y offset of Sample 4 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
2	31:28	**8x Sample3 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 3 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	27:24	**8x Sample3 Y Offset** Project: All Format: U0.4 Subpixel Y offset of Sample 3 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	23:20	**8x Sample2 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 2 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	19:16	**8x Sample2 Y Offset** Project: All Format: U0.4 Subpixel Y offset of Sample 2 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	15:12	**8x Sample1 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 1 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	11:8	**8x Sample1 Y Offset** Format: U0.4 Subpixel Y offset of Sample 1 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]

3DSTATE_SAMPLE_PATTERN		
	7:4	**8x Sample0 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 0 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	3:0	**8x Sample0 Y Offset** Project: All Format: U0.4 Subpixel Y offset of Sample 0 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
3	31:28	**4x Sample3 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 3 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]
	27:24	**4x Sample3 Y Offset** Project: All Format: U0.4 Subpixel Y offset of Sample 3 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]
	23:20	**4x Sample2 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 2 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]
	19:16	**4x Sample2 Y Offset** Project: All Format: U0.4 Subpixel Y offset of Sample 2 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]

3DSTATE_SAMPLE_PATTERN

DWord	Bit	Description
	15:12	**4x Sample1 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 1 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]
	11:8	**4x Sample1 Y Offset** Project: All Format: U0.4 Subpixel Y offset of Sample 1 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]
	7:4	**4x Sample0 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 0 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]
	3:0	**4x Sample0 Y Offset** Format: U0.4 Subpixel Y offset of Sample 0 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]
4	31:24	**Reserved** Project: All Format: MBZ
	23:20	**1x Sample0 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 0 relative to the UL pixel origin for 1x mode. Range: [0,0.9375]
	19:16	**1x Sample0 Y Offset** Project: All Format: U0.4 Subpixel Y offset of Sample 0 relative to the UL pixel origin for 1x mode. Range: [0,0.9375]

3DSTATE_SAMPLE_PATTERN		
	15:12	**2x Sample1 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 1 relative to the UL pixel origin for 2x mode. Range: [0,0.9375]
	11:8	**2x Sample1 Y Offset** Project: All Format: U0.4 Subpixel Y offset of Sample 1 relative to the UL pixel origin for 2x mode. Range: [0,0.9375]
	7:4	**2x Sample0 X Offset** Project: All Format: U0.4 Subpixel X offset of Sample 0 relative to the UL pixel origin for 2x mode. Range: [0,0.9375]
	3:0	**2x Sample0 Y Offset** Project: All Format: U0.4 Subpixel Y offset of Sample 0 relative to the UL pixel origin for 2x mode. Range: [0,0.9375]

3DSTATE_SAMPLER_PALETTE_LOAD0

3DSTATE_SAMPLER_PALETTE_LOAD0

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

Description
The 3DSTATE_SAMPLER_PALETTE_LOAD0 instruction is used to load 32-bit values into the first texture palette. The texture palette is used whenever a texture with a paletted format (containing "Px [palette0]") is referenced by the sampler.
This instruction is used to load all or a subset of the 256 entries of the first palette. Partial loads always start from the first (index 0) entry.

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: 1h 3DSTATE_NONPIPELINED Format: Opcode
	23:16	**3D Command Sub Opcode** Default Value: 02h 3DSTATE_SAMPLER_PALETTE_LOAD0 Format: Opcode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Format: =n Total Length = 1 + entryCount - 2 Value: [0,255] — Name: Range — Description: 1-256 Entries
1..n	31:0	**Entry** Format: PALETTE_ENTRY

3DSTATE_SAMPLER_PALETTE_LOAD1

3DSTATE_SAMPLER_PALETTE_LOAD1

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The 3DSTATE_SAMPLER_PALETTE_LOAD1 instruction is used to load 32-bit values into the second texture palette. The second texture palette is used whenever a texture with a paletted format (containing "Px...[palette1]") is referenced by the sampler.This instruction is used to load all or a subset of the 256 entries of the second palette. Partial loads always start from the first (index 0) entry.

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: 1h 3DSTATE Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 0Ch 3DSTATE_SAMPLER_PALETTE_LOAD1 Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1..n	31:24	**Palette Alpha[0:N-1]** Project: All Format: U8 Alpha channel loaded into the Nth entry of the texture color palette.
	23:16	**Palette Red[0:N-1]** Project: All Format: U8 Alpha channel loaded into the Nth entry of the texture color palette.

3DSTATE_SAMPLER_PALETTE_LOAD1		
	15:8	**Palette Green[0:N-1]** Project: All Format: U8 Alpha channel loaded into the Nth entry of the texture color palette.
	7:0	**Palette Blue[0:N-1]** Project: All Format: U8 Alpha channel loaded into the Nth entry of the texture color palette.

3DSTATE_SAMPLER_STATE_POINTERS_DS

3DSTATE_SAMPLER_STATE_POINTERS_DS		
Project:	CHV, BSW	
Source:	RenderCS	
Length Bias:	2	
The 3DSTATE_SAMPLER_STATE_POINTERS_DS command is used to define the location of DS SAMPLER_STATE table. Only some of the fixed functions utilize sampler state tables.		
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: 2Dh 3DSTATE_SAMPLER_STATE_POINTERS_DS Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:5	**Pointer to DS Sampler State** Project: All Format: DynamicStateOffset[31:5]SAMPLER_STATE*16 Specifies the 32-byte aligned address offset of the DS function's SAMPLER_STATE table. This offset is relative to the Dynamic State Base Address.
	4:0	**Reserved** Project: All Format: MBZ

3DSTATE_SAMPLER_STATE_POINTERS_GS

3DSTATE_SAMPLER_STATE_POINTERS_GS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The 3DSTATE_SAMPLER_STATE_POINTERS_GS command is used to define the location of GS SAMPLER_STATE table. Only some of the fixed functions utilize sampler state tables.

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: 2Eh 3DSTATE_SAMPLER_STATE_POINTERS_GS Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:5	**Pointer to GS Sampler State** Project: All Format: DynamicStateOffset[31:5]SAMPLER_STATE*16 Specifies the 32-byte aligned address offset of the GS function's SAMPLER_STATE table. This offset is relative to the Dynamic State Base Address.
	4:0	**Reserved** Project: All Format: MBZ

3DSTATE_SAMPLER_STATE_POINTERS_HS

3DSTATE_SAMPLER_STATE_POINTERS_HS		
Project:	CHV, BSW	
Source:	RenderCS	
Length Bias:	2	
The 3DSTATE_SAMPLER_STATE_POINTERS_HS command is used to define the location of HS SAMPLER_STATE table. Only some of the fixed functions utilize sampler state tables.		
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: 2Ch 3DSTATE_SAMPLER_STATE_POINTERS_HS Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:5	**Pointer to HS Sampler State** Project: All Format: DynamicStateOffset[31:5]SAMPLER_STATE*16 Specifies the 32-byte aligned address offset of the HS function's SAMPLER_STATE table. This offset is relative to the Dynamic State Base Address.
	4:0	**Reserved** Project: All Format: MBZ

3DSTATE_SAMPLER_STATE_POINTERS_PS

3DSTATE_SAMPLER_STATE_POINTERS_PS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The 3DSTATE_SAMPLER_STATE_POINTERS_PS command is used to define the location of PS SAMPLER_STATE table. Only some of the fixed functions utilize sampler state tables.

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: 2Fh 3DSTATE_SAMPLER_STATE_POINTERS_PS Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:5	**Pointer to PS Sampler State** Project: All Format: DynamicStateOffset[31:5]SAMPLER_STATE*16 Specifies the 32-byte aligned address offset of the PS function's SAMPLER_STATE table. This offset is relative to the Dynamic State Base Address.
	4:0	**Reserved** Project: All Format: MBZ

3DSTATE_SAMPLER_STATE_POINTERS_VS

3DSTATE_SAMPLER_STATE_POINTERS_VS		
Project:	CHV, BSW	
Source:	RenderCS	
Length Bias:	2	
The 3DSTATE_SAMPLER_STATE_POINTERS_VS command is used to define the location of VS SAMPLER_STATE table. Only some of the fixed functions utilize sampler state tables.		
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: 2Bh 3DSTATE_SAMPLER_STATE_POINTERS_VS Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:5	**Pointer to VS Sampler State** Project: All Format: DynamicStateOffset[31:5]SAMPLER_STATE*16 Specifies the 32-byte aligned address offset of the VS function's SAMPLER_STATE table. This offset is relative to the Dynamic State Base Address.
	4:0	**Reserved** Project: All Format: MBZ

3DSTATE_SBE

3DSTATE_SBE		
Project: CHV, BSW Source: RenderCS Length Bias: 2		
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: 1Fh 3DSTATE_SBE Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 02h Excludes DWord (0,1) Project: CHV, BSW Format: =n Total Length - 2
1	31:30	**Reserved** Format: MBZ
	29	**Force Vertex URB Entry Read Length** Project: CHV, BSW Format: Enable This field provides a work around override for the computation of SBE_INT::Vertex URB Entry Read Length. If asserted, 3DSTATE_SBE::Vertex URB Entry Read Length is be used directly. Otherwise, SBE_INT::Vertex URB Entry Read Length is computed normally.
	28	**Force Vertex URB Entry Read Offset** Project: CHV, BSW

3DSTATE_SBE

Format:	Enable

This field provides a work around override for the computation of SBE_INT::Vertex URB Entry Read Offset. If asserted, 3DSTATE_SBE::Vertex URB Entry Read Offset is be used directly. Otherwise, SBE_INT::Vertex URB Entry Read Offset is computed normally.

27:22 **Number of SF Output Attributes**

Project:	CHV, BSW
Format:	U6 count of attributes

Specifies the number of vertex attributes passed from the SF stage to the WM stage (does not include Position).

Value	Name
[0,32]	

21 **Attribute Swizzle Enable**

Format:	Enable

Enables the SF to perform swizzling on (up to the first 16) vertex attributes. If DISABLED, all vertex attributes are passed through.

20 **Point Sprite Texture Coordinate Origin**

This state controls how Point Sprite Texture Coordinates are generated (when enabled on a per-attribute basis by Point Sprite Texture Coordinate Enable).

Value	Name	Description
0h	UPPERLEFT	Top Left = (0,0,0,1)Bottom Left = (0,1,0,1)Bottom Right = (1,1,0,1)
1h	LOWERLEFT	Top Left = (0,1,0,1)Bottom Left = (0,0,0,1)Bottom Right = (1,0,0,1)

19 **Primitive ID Override Component W**

Project:	CHV, BSW
Format:	Enable

If set, the W component of output attribute selected by Primitive ID Override Attribute Select is overridden with the Primitive ID.

18 **Primitive ID Override Component Z**

Project:	CHV, BSW
Format:	Enable

If set, the Z component of output attribute selected by Primitive ID Override Attribute Select is overridden with the Primitive ID.

17 **Primitive ID Override Component Y**

Project:	CHV, BSW
Format:	Enable

If set, the Y component of output attribute selected by Primitive ID Override Attribute Select is overridden with the Primitive ID.

3DSTATE_SBE

	16	**Primitive ID Override Component X** Project: CHV, BSW Format: Enable If set, the X component of output attribute selected by Primitive ID Override Attribute Select is overridden with the Primitive ID.

	15:11	**Vertex URB Entry Read Length**
		Format: U5
		Specifies the amount of URB data read for each Vertex URB entry, in 256-bit register increments.
		Value: [1,16] — Name: (none)
		Programming Notes
		It is UNDEFINED to set this field to 0 indicating no Vertex URB data to be read.This field should be set to the minimum length required to read the maximum source attribute. The maximum source attribute is indicated by the maximum value of the enabled Attribute # Source Attribute if Attribute Swizzle Enable is set, Number of Output Attributes-1 if enable is not set.
		read_length = ceiling((max_source_attr+1)/2)

	10:5	**Vertex URB Entry Read Offset** Project: CHV, BSW Specifies the offset (in 256-bit units) at which Vertex URB data is to be read from the URB.
	4:0	**Primitive ID Override Attribute Select** Project: CHV, BSW Specifies which attribute is overridden w/ the Primitive ID **Programming Notes** Set all Primitive ID Override Component Select X/Y/Z/W to 0 to indicate there is no Primitive ID override.
2	31:0	**Point Sprite Texture Coordinate Enable** Format: Enable[32] When processing point primitives, the attributes from the incoming point vertex are typically copied to the point object corner vertices. However, if a bit is set in this field, the corresponding Attribute is selected as a Point Sprite Texture Coordinate, in which case each corner vertex is assigned a pre-defined texture coordinate as defined by the Point Sprite Texture Coordinate Origin state bit. Bit 0 corresponds to output Attribute 0.
3	31:0	**Constant Interpolation Enable** Format: Enable[32] This field is a bitmask containing a Constant Interpolation Enable bit for each corresponding attribute. If a bit is set, that attribute will undergo constant interpolation, and the corresponding WrapShortest Enable bits (if defined) will be ignored. If a bit is clear, components which are not enabled for WrapShortest interpolation (if defined) will be linearly interpolated.

3DSTATE_SBE_SWIZ

3DSTATE_SBE_SWIZ		
Project:	CHV, BSW	
Source:	RenderCS	
Length Bias:	2	
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: 51h 3DSTATE_SBE_SWIZ Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 9h Excludes DWord (0,1) Format: =n Total Length - 2
1..8	15:0	**Attribute** Format: SF_OUTPUT_ATTRIBUTE_DETAIL
9..10	63:60	**Attribute 15 Wrap Shortest Enables** Format: WRAP_SHORTEST_ENABLE
	59:56	**Attribute 14 Wrap Shortest Enables** Format: WRAP_SHORTEST_ENABLE
	55:52	**Attribute 13 Wrap Shortest Enables** Format: WRAP_SHORTEST_ENABLE
	51:48	**Attribute 12 Wrap Shortest Enables** Format: WRAP_SHORTEST_ENABLE
	47:44	**Attribute 11 Wrap Shortest Enables**

3DSTATE_SBE_SWIZ

		Format:	WRAP_SHORTEST_ENABLE
	43:40	**Attribute 10 Wrap Shortest Enables**	
		Format:	WRAP_SHORTEST_ENABLE
	39:36	**Attribute 09 Wrap Shortest Enables**	
		Format:	WRAP_SHORTEST_ENABLE
	35:32	**Attribute 08 Wrap Shortest Enables**	
		Format:	WRAP_SHORTEST_ENABLE
	31:28	**Attribute 07 Wrap Shortest Enables**	
		Format:	WRAP_SHORTEST_ENABLE
	27:24	**Attribute 06 Wrap Shortest Enables**	
		Format:	WRAP_SHORTEST_ENABLE
	23:20	**Attribute 05 Wrap Shortest Enables**	
		Format:	WRAP_SHORTEST_ENABLE
	19:16	**Attribute 04 Wrap Shortest Enables**	
		Format:	WRAP_SHORTEST_ENABLE
	15:12	**Attribute 03 Wrap Shortest Enables**	
		Format:	WRAP_SHORTEST_ENABLE
	11:8	**Attribute 02 Wrap Shortest Enables**	
		Format:	WRAP_SHORTEST_ENABLE
	7:4	**Attribute 01 Wrap Shortest Enables**	
		Format:	WRAP_SHORTEST_ENABLE
	3:0	**Attribute 00 Wrap Shortest Enables**	
		Format:	WRAP_SHORTEST_ENABLE

3DSTATE_SCISSOR_STATE_POINTERS

3DSTATE_SCISSOR_STATE_POINTERS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The 3DSTATE_SCISSOR_STATE_POINTERS command is used to define the location of the indirect SCISSOR_RECT state.

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: 0Fh 3DSTATE_SCISSOR_STATE_POINTERS Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:5	**Scissor Rect Pointer** Project: All Format: DynamicStateOffset[31:5]SCISSOR_RECT*16 Specifies the 32-byte aligned address offset of the SCISSOR_RECT state. This offset is relative to the **Dynamic State Base Address**.
	4:0	**Reserved** Project: All Format: MBZ

3DSTATE_SF

3DSTATE_SF

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

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: 13h 3DSTATE_SF Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 2h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:30	**Reserved** Format: MBZ
	29:12	**Line Width** Project: CHV, BSW Format: U11.7 Range: [0.0, 2047.9921875] Controls width of line primitives. Setting a Line Width of 0.0 specifies the rasterization of the "thinnest" (one-pixel-wide), non-antialiased lines. Note that this effectively overrides the effect of AAEnable (though the AAEnable state variable is not modified).

3DSTATE_SF

DWord	Bit	Description
		Programming Notes Software must not program a value of 0.0 when running in MSRASTMODE_ON_xxx modes - zero-width lines are not available when multisampling rasterization is enabled. **Restriction** / **Project** Line widths higher than 40.0 will appear to be thinner than the programmed line width at certain angles. Software must expose the line widths higher than 40 to APIs only if the loss in quality is acceptable. — CHV, BSW Software must set the "WM_CHIKEN1" register bits "1 and 3" when rendering the AA lines higher than Line width of "40.0". — CHV, BSW
	11	**Legacy Global Depth Bias Enable** Format: Enable Enables the SF to use the Global Depth Offset Constant state unmodified. If this bit is not set, the SF will scale the Global Depth Offset Constant as described in section Error! Reference source not found. of this document. **Programming Notes** This bit should be set whenever non zero depth bias (Slope, Bias) values are used. Setting this bit may have some degradation of performance for some workloads.
	10	**Statistics Enable** Project: All Format: Enable If ENABLED, this FF unit will increment CL_PRIMITIVES_COUNT on behalf of the CLIP stage. If DISABLED, CL_PRIMITIVES_COUNT will be left unchanged. **Programming Notes** This bit should be set whenever clipping is enabled and the Statistics Enable bit is set in CLIP_STATE. It should be cleared if clipping is disabled or Statistics Enable in CLIP_STATE is clear.
	9:2	**Reserved** Format: MBZ
	1	**Viewport Transform Enable** Format: Enable This bit controls the Viewport Transform function.
	0	**Reserved** Format: MBZ
2	31:29	**Reserved** Format: MBZ
	28	**Reserved**
	27:18	**Reserved**

3DSTATE_SF

DWord	Bit	Description
	17:16	**Line End Cap Antialiasing Region Width** Format: U2 This field specifies the distances over which the coverage of anti-aliased line end caps are computed. Value / Name / Description: 0h / 0.5 pixels / 0.5 pixels 1h / 1.0 pixels / 1.0 pixels 2h / 2.0 pixels / 2.0 pixels 3h / 4.0 pixels / 4.0 pixels
	15	**Reserved** Format: MBZ
	14	**Reserved** Format: MBZ
	13	**Reserved**
	12	**Reserved**
	11:0	**Reserved** Format: MBZ
3	31	**Last Pixel Enable** Format: Enable If ENABLED, the last pixel of a diamond line will be lit. This state will only affect the rasterization of Diamond lines (will not affect wide lines or anti-aliased lines). **Programming Notes** Last pixel is applied to all lines of a LINELIST, and only the last line of a LINESTRIP.
	30:29	**Triangle Strip/List Provoking Vertex Select** Format: 0-based vertex index Selects which vertex of a triangle (in a triangle strip or list primitive) is considered the "provoking vertex". Used for flat shading of primitives.Does current implementation send provoking vertex first? Value / Name: 0h / 0 1h / 1 2h / 2 3h / Reserved
	28:27	**Line Strip/List Provoking Vertex Select** Project: All Format: 0-based vertex index Selects which vertex of a line (in a line strip or list primitive) is considered the "provoking vertex".

3DSTATE_SF

Value	Name	Description
0h	0	Vertex 0
1h	1	Vertex 1
2h	Reserved	Reserved
3h	Reserved	Reserved

26:25 **Triangle Fan Provoking Vertex Select**

Format:	0-based vertex index

Selects which vertex of a triangle (in a triangle fan primitive) is considered the "provoking vertex".

Value	Name
0h	0
1h	1
2h	2
3h	Reserved

24:15 **Reserved**

Format:	MBZ

14 **AA Line Distance Mode**

Format:	U1

This bit controls the distance computation for antialiased lines.

Value	Name	Description
1h	AALINEDISTANCE_TRUE	True distance computation. This is the normal setting which should yield WHQL compliance.

13 **Smooth Point Enable**

Format:	Enable
Double Buffer Armed By:	Enables logic to draw smooth OGL Points

Programming Notes
If Enabled, SF will treat points in the same fashion that AA lines are processed

12 **Vertex Sub Pixel Precision Select**

Format:	U1

Selects the number of fractional bits maintained in the vertex data

Value	Name	Description
0h	Disable	8 sub pixel precision bits maintained
1h	Enable	4 sub pixel precision bits maintained

3DSTATE_SF

	11	**Point Width Source** Controls whether the point width passed on the vertex or from state is used for rendering point primitives.
	10:0	**Point Width**

Value	Name	Description
0h	Vertex	Use Point Width on Vertex
1h	State **[Default]**	Use Point Width from State

Format:	U8.3

Range: [0.125, 255.875] pixels
This field specifies the size (width) of point primitives in pixels. This field is overridden (though not overwritten) whenever point width information is passed in the FVF

3DSTATE_SO_BUFFER

3DSTATE_SO_BUFFER

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

Programming Notes	Project
Foreach SO Buffer, the 3DSTATE_SO_BUFFER must only be sent once prior to each 3DPRIMITIVE command.	CHV, BSW

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: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 18h 3DSTATE_SO_BUFFER Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 6h Excludes DWord (0,1) Format: =n Total Length - 2
1	31	**SO Buffer Enable** Format: Enable If set, stream output to SO Buffer is enabled, , if 3DSTATE_STREAMOUT::SO Function ENABLE is also enabled..If clear, the SO Buffer is considered "not bound" and effectively treated as a zero-length buffer for the purposes of SO output and overflow detection. If an enabled stream's Stream to Buffer Selects includes this buffer it is by definition an overflow condition. That stream will cause no writes to occur, and only SO_PRIM_STORAGE_NEEDED[<stream>] will increment.

3DSTATE_SO_BUFFER		
	30:29	**SO Buffer Index** Format: U2 Specifies which of the four SO Buffers is being defined.
	28:22	**SO Buffer Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for the SO buffer.
	21	**Stream Offset Write Enable** Format: Enable When set, this field allows the hardware to write SO_WRITE_OFFSET[Buffer#] as specified in the Stream Offset field. **Programming Notes** The field is operates irrespective of whether SO Buffer Enable is set or clear.
	20	**Stream Output Buffer Offset Address Enable** Format: Enable When set, this field allows the hardware to read/write the stream output buffer offset as specified in the "Stream Output Buffer Offset Address" field. **Programming Notes** The field is operates irrespective of whether SO Buffer Enable is set or clear.
	19:12	**Reserved** Format: MBZ
	11:0	**Reserved** Format: MBZ
2..3	63:48	**Reserved** Format: MBZ
	47:2	**Surface Base Address** Format: GraphicsAddress[47:2]SurfaceBase This field specifies the starting DWord address of the buffer in Graphics Memory.
	1:0	**Reserved** Format: MBZ
4	31:30	**Reserved** Format: MBZ
	29:0	**Surface Size** Format: U30-1 This field specifies the size of buffer in number DWords minus 1 of the buffer in Graphics Memory.

3DSTATE_SO_BUFFER		
5..6	63:48	**Reserved** Format: MBZ
	47:2	**Stream Output Buffer Offset Address** Format: GraphicsAddress[47:2]OutputBuffer This field specifies the high 16 bits of address of the buffer in Graphics Memory where the Stream Output Buffer Offset is stored when all the data has been written. It is also used to fetch the stream Output buffer Offset when needed.
	1:0	**Reserved** Format: MBZ
7	31:0	**Stream Offset** This field specifies the Offset in stream output buffer to start at, or whether to append to the end of an existing buffer. The Offset must be DWORD aligned. If Stream Offset is equal to 0xFFFFFFFF then load the value at the Stream Output Buffer Offset address into SO_WRITE_OFFSET[Buffer#]. Otherwise, SO_WRITE_OFFSET[Buffer#] = Stream Offset.

Value	Name	Description
FFFFFFFFh		Load the value at the Stream Output Buffer Offset address into MMIO_SO_OFFSET[Buffer#].
xxxxxxxx xxxxxxxx xxxxxxxx xxxxxx00b		MMIO_SO_OFFSET[Buffer#] = Stream Offset

3DSTATE_SO_DECL_LIST

3DSTATE_SO_DECL_LIST

Project: CHV, BSW
Source: RenderCS
Length Bias: 2

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: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 17h 3DSTATE_SO_DECL_LIST Format: OpCode
	15:9	**Reserved** Format: MBZ
	8:0	**DWord Length** Format: =n Total Length - 2 Value: [1,257] — Name: Excludes DWORD (0,1) 0-128 Entries — Description: Value = 2 * (# of SO_DECL quads) + 1
1	31:16	**Reserved** Format: MBZ
	15:12	**Stream to Buffer Selects [3]** Format: U4 bitmask Identifies to which SO Buffers stream 3 outputs. See Stream To Buffer Selects [0] field description.
	11:8	**Stream to Buffer Selects [2]** Format: U4 bitmask Identifies to which SO Buffers stream 2 outputs. See Stream To Buffer Selects [0] field description.

3DSTATE_SO_DECL_LIST

	7:4	**Stream to Buffer Selects [1]** Format: U4 bitmask Identifies to which SO Buffers stream 1 outputs. See Stream To Buffer Selects [0] field description.
	3:0	**Stream to Buffer Selects [0]** Format: U4 bitmask Identifies to which SO Buffers stream 0 outputs (irrespective of whether those buffers are enabled via 3DSTATE_STREAMOUT). Software is required to scan the SO_DECL list in order to provide this summary information. Note: For "inactive" streams, software must program this field to all zero (no buffers written to) and the corresponding Num Entries field to zero (no valid SO_DECLs). Value / Name: 1xxxb: SO Buffer 3 x1xxb: SO Buffer 2 xx1xb: SO Buffer 1 xxx1b: SO Buffer 0
2	31:24	**Num Entries [3]** Format: U8 #entries Specifies the number of valid SO_DECL entries for Stream 3. (See notes in Num Entries [0] field description). Value / Name: [0,128]: entries
	23:16	**Num Entries [2]** Format: U8 #entries Specifies the number of valid SO_DECL entries for Stream 2. (See notes in Num Entries [0] field description). Value / Name: [0,128]: entries
	15:8	**Num Entries [1]** Format: U8 #entries Specifies the number of valid SO_DECL entries for Stream 1. (See notes in Num Entries [0] field description). Value / Name: [0,128]: entries

<table>
<tr><th colspan="3">3DSTATE_SO_DECL_LIST</th></tr>
<tr><td></td><td>7:0</td><td>Num Entries [0]
Format: U8 #entries
Specifies the number of valid SO_DECL entries for Stream 0.Note that the SO_DECLs are programmed in groups of four (one SO_DECL for each of the four streams). Therefore the number of 2-DWord groups of SO_DECLs supplied in this command is derived from the stream(s) with the most valid SO_DECLs. The NumEntries value specific to each stream will indicate how many SO_DECLS are valid for that particular stream. Any trailing invalid SO_DECLs supplied for streams with fewer valid SO_DECLs will be ignored. It is legal to specify Num Entries = 0 for all four streams simultaneously. In this case there will be no SO_DECLs included in the command (only DW 0-2). Note that all Stream to Buffer Selects bits must be zero in this case (as no streams produce output).
<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,128]</td><td>entries</td></tr></table></td></tr>
<tr><td>3..n</td><td>63:0</td><td>Entry
Format: SO_DECL_ENTRY [CHV, BSW]</td></tr>
</table>

3DSTATE_STENCIL_BUFFER

3DSTATE_STENCIL_BUFFER

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

Description	Project
This command sets the surface state of the separate stencil buffer, delivered as a pipelined state command. However, the state change pipelining isn't completely transparent (see restriction below).	
WM HW will internally manage the draining pipe and flushing of the caches when this command is issued. The PIPE_CONTROL restrictions are removed.	CHV, BSW

Programming Notes	Project
Restriction: Prior to changing Depth/Stencil Buffer state (i.e., any combination of 3DSTATE_DEPTH_BUFFER, 3DSTATE_CLEAR_PARAMS, 3DSTATE_STENCIL_BUFFER, 3DSTATE_HIER_DEPTH_BUFFER) SW must first issue a pipelined depth stall (PIPE_CONTROL with Depth Stall bit set, followed by a pipelined depth cache flush (PIPE_CONTROL with Depth Flush Bit set, followed by another pipelined depth stall (PIPE_CONTROL with Depth Stall Bit set), unless SW can otherwise guarantee that the pipeline from WM onwards is already flushed (e.g., via a preceding MI_FLUSH).	
The stencil buffer is always Tile-W	CHV, BSW

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: 06h 3DSTATE_STENCIL_BUFFER Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**Dword Length** Format: =n Total Length - 2

3DSTATE_STENCIL_BUFFER

<table>
<tr><td></td><td></td><td><table><tr><th>Value</th><th>Name</th><th>Project</th></tr><tr><td>3h</td><td>Excludes Dword (0,1) [Default]</td><td>CHV, BSW</td></tr></table></td></tr>
<tr><td>1</td><td>31</td><td>Stencil Buffer Enable<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>U1</td></tr></table>When set indicates that there is a valid stencil buffer.<table><tr><th>Programming Notes</th></tr><tr><td>This bit should be "0" if Depth buffer surface format is D16_UNORM OR Depth buffer surface type is NULL.</td></tr></table></td></tr>
<tr><td></td><td>30:29</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>28:22</td><td>Stencil Buffer Object Control State<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>MEMORY_OBJECT_CONTROL_STATE</td></tr></table>Specifies the memory object control state for the stencil buffer.</td></tr>
<tr><td></td><td>21:17</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>16:0</td><td>Surface Pitch<table><tr><td>Format:</td><td>U17-1 Pitch in Bytes</td></tr></table>This field specifies the pitch of the stencil buffer in (#Bytes - 1).<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>[127, 1FFFFh]</td><td></td><td>corresponding to [128B, 128KB]also restricted to a multiple of 128B</td></tr></table><table><tr><th>Programming Notes</th></tr><tr><td>Since this surface is tiled, the pitch specified must be a multiple of the tile pitch, in the range [128B, 128KB].</td></tr><tr><td>The pitch must be set to 2x the value computed based on width, as the stencil buffer is stored with two rows interleaved. For details on the separate stencil buffer storage format in memory, see GPU Overview (vol1a), Memory Data Formats, Surface Layout, 2D Surfaces, Stencil Buffer Layout (section 8.20.4.8).</td></tr></table></td></tr>
<tr><td>2..3
Project:
CHV, BSW</td><td>63:0</td><td>Surface Base Address<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>GraphicsAddress[63:0]Stencil_Buffer</td></tr></table>This field specifies the address of the buffer in Graphics Memory.<table><tr><th>Programming Notes</th></tr><tr><td>The Stencil Buffer can only be mapped to Main Memory (uncached).</td></tr></table></td></tr>
</table>

3DSTATE_STENCIL_BUFFER

DWord: 4
Project: CHV, BSW

31:15 Reserved

Project:	CHV, BSW
Format:	MBZ

14:0 Surface QPitch

Project:	CHV, BSW
Format:	QPitch[16:2]

Description	Project
This field specifies the distance in rows between array slices. It is used only in the following cases: • **Surface Array** is enabled *OR* • **Number of Mulitsamples** is not NUMSAMPLES_1 and **Multisampled Surface Storage Format** set to MSFMT_MSS *OR* • **Surface Type** is SURFTYPE_CUBE	CHV, BSW

Value	Name	Description
[4h,1FFFCh]		in multiples of 4 (low 2 bits missing)

Programming Notes
This field must be set to an integer multiple of 8 (QPitch[2] MBZ) Software must ensure that this field is set to a value sufficiently large such that the array slices in the surface do not overlap. Refer to the Memory Data Formats section for information on how surfaces are stored in memory.

3DSTATE_STREAMOUT

3DSTATE_STREAMOUT

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command contains pipelined state required by the SOL unit.

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: 1Eh 3DSTATE_STREAMOUT Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 3h Excludes DWord (0,1) Project: CHV, BSW Format: =n Total Length - 2
1	31	**SO Function Enable** Project: All Format: U1 If set, the SO function is enabled. Vertex data will be streamed out to memory (subject to overflow detection) as controlled by the various SO-related state variables. If clear, the SO function is disabled, and therefore no vertex data will be streamed out to memory. However, the Rendering Disable and Render Stream Select fields will still be used to determine which vertices (if any) are forwarded down the pipeline for (possible) rendering.

3DSTATE_STREAMOUT

	30	**API Rendering Disable** Project: CHV, BSW Format: U1 If set, Indicates the API wants the SO stage not to forward any topologies down the pipeline. If clear, Indicates the API wants the SO stage to forward topologies associated with **Render Stream Select** down the pipeline. This bit is used even if **SO Function Enable** is DISABLED. **Programming Notes** The SOL unit generates an SOL_INT::Render_Enable which ultimately controls whether rendering occurs or not.
	29	**Reserved** Project: All Format: MBZ
	28:27	**Render Stream Select** Project: All Format: U2 **Description** This field specifies which stream has been selected to be forwarded down the pipeline for possible rendering. Topologies from other streams will not be passed down the pipeline. If Rendering Disable is set, this field is ignored, as no topologies are sent down the pipeline. SO Function Enable must also be ENABLED in order for thiis field to select a stream for rendering. When **SO Function Enable** is DISABLED and Rendering Disable is cleared (i.e., rendering is enabled), StreamID is ignored downstream of the SO stage, allowing any stream to be rendered.
	26	**Reorder Mode** Project: All This bit controls how vertices of triangle objects in TRISTRIP[_ADJ] and TRISTRIP_REV are reordered for the purposes of stream-out only (does not impact rendering). See table in Input Buffering. **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.
	25	**SO Statistics Enable** Project: All Format: Enable This bit controls whether StreamOutput statistics register(s) can be incremented.

3DSTATE_STREAMOUT

DWord	Bit	Description
		(continued)

Value	Name	Description	Project
0h	Disable	SO_NUM_PRIMS_WRITTEN[0..3] and SO_PRIM_STORAGE_NEEDED[0..3] registers cannot increment.	All
1h	Enable	SO_NUM_PRIMS_WRITTEN[0..3] and SO_PRIM_STORAGE_NEEDED[0..3] registers can increment.	All

24:23 Force Rendering

Project:	CHV, BSW

This field provides a work around override for the computation of SOL_INT::Render_Enable

Value	Name	Description	Project
0h	Normal	SOL_INT::Render_Enable is computed normally	All
1h	Resreved		All
2h	Force_Off	Forces the rendering to be disabled.	All
3h	Force_on	Forces the rendering to be enabled.	All

22:12 Reserved

Project:	All
Format:	MBZ

11:8 Reserved

Project:	CHV, BSW
Format:	MBZ

7:0 Reserved

Project:	All
Format:	MBZ

DWord 2

31:30 Reserved

Project:	All
Format:	MBZ

29 Stream 3 Vertex Read Offset

Project:	All
Format:	U1 count of 256-bit units

Specifies amount of data to skip over before reading back Stream 3 vertex data. (See **Stream 0 Vertex Read Offset**)

28:24 Stream 3 Vertex Read Length

Project:	All
Format:	U5-1 count of 256-bit units

(See Stream 0 Vertex Read Length)

23:22 Reserved

3DSTATE_STREAMOUT

		Project: All Format: MBZ
	21	**Stream 2 Vertex Read Offset** Project: All Format: U1 count of 256-bit units Specifies amount of data to skip over before reading back Stream 2 vertex data. (See Stream 0 Vertex Read Offset)
	20:16	**Stream 2 Vertex Read Length** Project: All Format: U5-1 count of 256-bit units
	15:14	**Reserved** Project: All Format: MBZ
	13	**Stream 1 Vertex Read Offset** Project: All Format: U1 count of 256-bit units Specifies amount of data to skip over before reading back Stream 1 vertex data. (See Stream 0 Vertex Read Offset)
	12:8	**Stream 1 Vertex Read Length** Project: All Format: U5-1 count of 256-bit units (See Stream 0 Vertex Read Length)
	7:6	**Reserved** Project: All Format: MBZ
	5	**Stream 0 Vertex Read Offset** Project: All Format: U1 count of 256-bit units Specifies amount of data to skip over before reading back Stream 0 vertex data. Must be zero if the GS is enabled and the Output Vertex Size field in 3DSTATE_GS is programmed to 0 (i.e., one 16B unit).
	4:0	**Stream 0 Vertex Read Length** Project: All Format: U5-1 count of 256-bit units

3DSTATE_STREAMOUT

<table>
<tr><td></td><td></td><td>Specifies amount of vertex data to read back for Stream 0 vertices, starting at the Stream 0 Vertex Read Offset location. Maximum readback is 17 256-bit units (34 128-bit vertex attributes). Read data past the end of the valid vertex data has undefined contents, and therefore shouldn't be used to source stream out data. Must be zero (i.e., read length = 256b) if the GS is enabled and the Output Vertex Size field in 3DSTATE_GS is programmed to 0 (i.e., one 16B unit).</td></tr>
<tr><td rowspan="4">3
Project:
CHV, BSW</td><td>31:28</td><td>Reserved
Project: CHV, BSW
Format: MBZ</td></tr>
<tr><td>27:16</td><td>Buffer 1 Surface Pitch
Project: CHV, BSW</td></tr>
<tr><td>15:12</td><td>Reserved
Project: CHV, BSW
Format: MBZ</td></tr>
<tr><td>11:0</td><td>Buffer 0 Surface Pitch
Project: CHV, BSW
Format: U12 pitch in Bytes
This field specifies the pitch of the SO buffer in #Bytes.
<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,2048]</td><td>Must be 0 or a multiple of 4 Bytes.</td></tr></table>
<table><tr><th>Programming Notes</th></tr><tr><td>A Surface Pitch of 0 indicates an un-bound buffer. No writes are performed. Surface Base Address is ignored.</td></tr></table></td></tr>
<tr><td rowspan="4">4
Project:
CHV, BSW</td><td>31:28</td><td>Reserved
Project: CHV, BSW
Format: MBZ</td></tr>
<tr><td>27:16</td><td>Buffer 3 Surface Pitch
Project: CHV, BSW
Format: U12</td></tr>
<tr><td>15:12</td><td>Reserved
Project: CHV, BSW
Format: MBZ</td></tr>
<tr><td>11:0</td><td>Buffer 2 Surface Pitch
Project: CHV, BSW
Format: U12</td></tr>
</table>

3DSTATE_TE

3DSTATE_TE		
Project:	CHV, BSW	
Source:	RenderCS	
Length Bias:	2	

For [CHV, BSW], the state used by TE 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: 1Ch 3DSTATE_TE Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 2h Excludes DWord (0,1) Project: All Format: =n Total Length - 2
1	31:20	**Reserved** Project: All Format: MBZ
	19	**Reserved** Project: CHV, BSW Format: MBZ
	18:16	**Reserved** Project: CHV, BSW Format: MBZ

3DSTATE_TE

	15:14	**Reserved** Project: All Format: MBZ
	13:12	**Partitioning** Project: All Format: U2 This field specifies how edges are partitioned based on tessellation factor.

Value	Name	Description	Project
0h	INTEGER	Outside/inside edges are divided into an integer number of equal-sized segments.	All
1h	ODD_FRACTIONAL	Outside/inside edges are divided into an odd number of possibly-unequal-sized segments.	All
2h	EVEN_FRACTIONAL	Outside/inside edges are divided into an even number of possibly-unequal-sized segments.	All

	11:10	**Reserved** Project: All Format: MBZ
	9:8	**Output Topology** Project: All Format: U2 This field specifies which primitive types are to be output.

Value	Name	Description	Project
0h	POINT	Points are output (as POINTLIST topologies)	All
1h	LINE	Lines are output (as LINESTRIP topologies). Only valid if ISOLINE domain is selected.	All
2h	TRI_CW	Clockwise-ordered triangles are output (either as TRISTRIP, TRISTRIP_REV or TRILIST topologies). Not valid if ISOLINE domain is selected.	All
3h	TRI_CCW	Count-clockwise-ordered triangles are output (either as TRISTRIP, TRISTRIP_REV or TRILIST topologies). Not valid if ISOLINE domain is selected.	All

	7:6	**Reserved** Project: All Format: MBZ
	5:4	**TE Domain** Project: All Format: U2 This field specifies which type of domain is to be tessellated.

3DSTATE_TE

		<table><tr><th>Value</th><th>Name</th><th>Description</th><th>Project</th></tr><tr><td>0h</td><td>QUAD</td><td>2D (U, V) domain is tessellated</td><td>All</td></tr><tr><td>1h</td><td>TRI</td><td>Triangular (U, V, W) domain is tessellated</td><td>All</td></tr><tr><td>2h</td><td>ISOLINE</td><td>2D (U, V) domain is tessellated.</td><td>All</td></tr></table>
	3	**Reserved** <table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table>
	2:1	**TE Mode** <table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>U2</td></tr></table>When TE Enable is ENABLED, this field specifies the overall operation of the TE stage.This field is ignored if TE Enable is DISABLED.<table><tr><th>Value</th><th>Name</th><th>Description</th><th>Project</th></tr><tr><td>0h</td><td>HW_TESS</td><td>Normal HW Tessellation Mode. The TessFactors are read from the patch URB entry, and are used to perform fixed-function hardware tessellation of the specified domain.</td><td>All</td></tr><tr><td>1h</td><td>SW_TESS</td><td>Software Tessellation Mode. The TE unit will pass down HS-thread-generated tessellated domain points instead of generating them itself from TessFactors. The TE unit will read the Domain Point Count and Domain Point Buffer Starting Address fields from the patch header, and if the count is 0 it will consider the patch culled and discard it. Otherwise the address is used to start fetching DOMAIN_POINT structures from memory and passing them down the pipeline to DS.</td><td>CHV, BSW</td></tr></table>
	0	**TE Enable** <table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>Enable</td></tr></table>If ENABLED, the TE stage will perform tessellation processing on incoming patch primitives. The TE Mode field determines how this tessellation operation proceeds.If DISABLED, the TE goes into pass-through mode. All other state fields are ignored.<table><tr><th>Programming Notes</th></tr><tr><td>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.</td></tr></table>
2	31:0	**Maximum Tessellation Factor Odd** <table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>IEEE_Float</td></tr></table>This field specifies the maximum TessFactor for ODD_FRACTIONAL partitioning when in HW_TESS mode.

3DSTATE_TE

Value	Name	Description
[427c0000h,427c0000h]	63 **[Default]**	Per API Spec, For normal operation software should set this value to 63.0

Programming Notes
Note that ISOLINE's LineDensity TF is always subjected to INTEGER partitioning regardless of the Partitioning state.

DWord	Bit	Description
3	31:0	**Maximum Tessellation Factor Not Odd**

Project:	All
Format:	IEEE_Float

This field specifies the maximum TessFactor for EVEN_FRACTIONAL or INTEGER partitioning when in HW_TESS mode.

Value	Name	Description
[42800000h,42800000h]	64 **[Default]**	Per API Spec, For normal operation software should set this value to 64.0

Programming Notes
Note that ISOLINE's LineDensity TF is always subjected to INTEGER partitioning regardless of the Partitioning state.

3DSTATE_URB_DS

3DSTATE_URB_DS

Project:	CHV, BSW
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**	
		Project:	All
		Format:	MBZ
	7:0	**DWord Length**	
		Default Value:	0h DWORD_COUNT_n
		Project:	All

3DSTATE_URB_DS

<table>
<tr><td></td><td></td><td>

Format:	=n

</td></tr>
<tr><td rowspan="3">1</td><td>31:25</td><td>

DS URB Starting Address

Project:	CHV, BSW
Format:	U7

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

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

</td></tr>
<tr><td>24:16</td><td>

DS URB Entry Allocation Size

Project:	All
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]	

</td></tr>
<tr><td>15:0</td><td>

DS Number of URB Entries

Project:	All

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
[0,1536]	
[0,384]	

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"

</td></tr>
</table>

3DSTATE_URB_GS

3DSTATE_URB_GS

Project:	CHV, BSW
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**	
		Project:	All
		Format:	MBZ
	7:0	**DWord Length**	
		Default Value:	0h DWORD_COUNT_n
		Project:	All

<table>
<tr><th colspan="3">3DSTATE_URB_GS</th></tr>
<tr><td></td><td></td><td><table><tr><td>Format:</td><td>=n</td></tr></table></td></tr>
<tr><td>1</td><td>31:25</td><td>GS URB Starting Address
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>U7</td></tr></table>
Offset from the start of the URB memory where GS starts its allocation, specified in multiples of 8 KB.
<table><tr><th>Value</th><th>Name</th><th>Project</th><th>Exists If</th></tr>
<tr><td>[0,48]</td><td></td><td></td><td>Device[SliceCount] == 1</td></tr>
<tr><td>[4,48]</td><td></td><td></td><td>Device[SliceCount] GT 1</td></tr></table></td></tr>
<tr><td></td><td>24:16</td><td>GS URB Entry Allocation Size
<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>U9-1 512-bit units</td></tr></table>
Specifies the length of each URB entry owned by GS. This field is always used (even if GS Function Enable is DISABLED).</td></tr>
<tr><td></td><td>15:0</td><td>GS Number of URB Entries
<table><tr><td>Project:</td><td>All</td></tr></table>
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).
<table><tr><th>Value</th><th>Name</th><th>Project</th></tr>
<tr><td>[0,960]</td><td></td><td>CHV, BSW</td></tr>
<tr><td>[0,256]</td><td></td><td>CHV, BSW</td></tr></table>
<table><tr><th>Programming Notes</th></tr>
<tr><td>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.</td></tr>
<tr><td>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"</td></tr></table></td></tr>
</table>

3DSTATE_URB_HS

3DSTATE_URB_HS

Project:	CHV, BSW
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** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n

3DSTATE_URB_HS

1	31:25	**HS URB Starting Address** Project: CHV, BSW Format: U7 Offset from the start of the URB memory where HS starts its allocation, specified in multiples of 8 KB.
	24:16	**HS URB Entry Allocation Size** Project: All 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** Project: All 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"

HS URB Starting Address values:

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

HS Number of URB Entries values:

Value	Name	Project
[0,504]		CHV, BSW
[0,80]		CHV, BSW

3DSTATE_URB_VS

3DSTATE_URB_VS

Project:	CHV, BSW
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 Project: All Format: =n

3DSTATE_URB_VS

1	31:25	**VS URB Starting Address** Project: CHV, BSW Format: U7 Offset from the start of the URB memory where VS starts its allocation, specified in multiples of 8 KB.
	24:16	**VS URB Entry Allocation Size** Project: All 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).
	15:0	**VS Number of URB Entries** Project: All 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).

VS URB Starting Address (31:25)

Project:	CHV, BSW
Format:	U7

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

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

VS URB Entry Allocation Size (24:16)

Project:	All
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).

Value	Name	Project
[0,9]		CHV, BSW

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.

VS Number of URB Entries (15:0)

Project:	All
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
[64,2560]	
[34,640]	

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"
When tessellation is enabled, the VS Number of URB Entries must be greater than or equal to 192.

3DSTATE_VERTEX_BUFFERS

3DSTATE_VERTEX_BUFFERS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

Description
This command is used to specify VB state used by the VF function.
[CHV, BSW]: Can specify from 1 to 33 VBs.
The VertexBufferID field within a VERTEX_BUFFER_STATE structure indicates the specific VB. If a VB definition is not included in this command, its associated state is left unchanged and is available for use if previously defined.

Programming Notes
It is possible to have individual vertex elements sourced completely from generated ID values and therefore not require any vertex buffer accesses for that vertex element. In this case, VF function will simply ignore the VB state associated with that vertex element. If all enabled vertex elements have this characteristic, no VBs are required to process 3DPRIMITIVE commands. For example, this might arise when the user wants to perform all data lookups in the first shader, so only generated index values need to be passed down to it. In this extreme case, SW would not need to program any VB state, and therefore not need to issue any 3DSTATE_VERTEX_BUFFERS commands.
For any 3DSTATE_VERTEX_BUFFERS command, at least one VERTEX_BUFFER_STATE structure must be included.
VERTEX_BUFFER_STATE structures are 4 DWords for both VERTEXDATA buffers and INSTANCEDATA buffers.
Inclusion of partial VERTEX_BUFFER_STATE structures is UNDEFINED.
The order in which VBs are defined within this command can be arbitrary, though a vertex buffer must be defined only once in any given command (otherwise operation is UNDEFINED).

DWord	Bit	Description	
0	31:29	**Command Type**	
		Default Value:	03h GFXPIPE
		Format:	Opcode
	28:27	**Command SubType**	
		Default Value:	3h 3D
		Format:	Opcode
	26:24	**3D Command Opcode**	
		Default Value:	0h 3DSTATE_VERTEX_BUFFERS
		Format:	Opcode
	23:16	**3D Command Sub Opcode**	
		Default Value:	08h 3DSTATE_VERTEX_BUFFERS
		Format:	Opcode

3DSTATE_VERTEX_BUFFERS

	15:8	**Reserved**
	7:0	**DWord Length** Format: =n n = 4b-1 (where b = # of buffer states included) Value / Name / Project: 3 — DWORD_COUNT_n **[Default]** [3,131] — 1-33 Buffers — CHV, BSW
1..n	127:0	**Vertex Buffer State** Format: VERTEX_BUFFER_STATE [CHV, BSW]

3DSTATE_VERTEX_ELEMENTS

3DSTATE_VERTEX_ELEMENTS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

Description
This is a variable-length command used to specify the active vertex elements. Each VERTEX_ELEMENT_STATE structure contains a Valid bit which determines which elements are used.
[CHV, BSW]: Up to 34 elements.

Programming Notes
[CHV, BSW]: At least one VERTEX_ELEMENT_STATE structure must be included.
The 3DSTATE_VERTEX_ELEMENTS must not be programmed more than once before each 3DPRIMITIVE command.
Inclusion of partial VERTEX_ELEMENT_STATE structures is UNDEFINED.
[CHV, BSW]: SW must ensure that at least one vertex element is defined prior to issuing a 3DPRIMTIVE command, or operation is UNDEFINED.
[CHV, BSW]: There are no 'holes' allowed in the destination vertex: NOSTORE components must be overwritten by subsequent components unless they are the trailing DWords of the vertex. Software must explicitly chose some value (probably 0) to be written into DWords that would otherwise be 'holes'.
[CHV, BSW]: Within a VERTEX_ELEMENT_STATE structure, if a Component Control field is set to something other than VFCOMP_STORE_SRC, no higher-numbered Component Control fields may be set to VFCOMP_STORE_SRC. In other words, only trailing components can be set to something other than VFCOMP_STORE_SRC.
[CHV, BSW]: See additional restrictions listed in the command fields and VERTEX_ELEMENT_STATE description.
[CHV, BSW]: Element[0] must be valid.
[CHV, BSW]: All elements must be valid from Element[0] to the last valid element. (I.e. if Element[2] is valid then Element[1] and Element[0] must also be valid).
[CHV, BSW]: The pitch between elements packed in the URB will always be 128 bits.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 03h GFXPIPE Format: Opcode
	28:27	**Command SubType** Default Value: 3h 3D Format: Opcode

3DSTATE_VERTEX_ELEMENTS

	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_VERTEX_ELEMENTS Format: Opcode
	23:16	**3D Command Sub Opcode** Default Value: 09h 3DSTATE_VERTEX_ELEMENTS Format: Opcode
	15:8	**Reserved**
	7:0	**DWord Length** Format: =n Vertex Element Count = (DWord Count + 1) / 2
1..n	63:0	**Element** Format: VERTEX_ELEMENT_STATE

Value	Name	Description	Project
1	DWORD_COUNT_n **[Default]**	excludes DWords 0,1	
[1,67]	Range	1-34 Elements	CHV, BSW

3DSTATE_VF

3DSTATE_VF		
Project:	CHV, BSW	
Source:	RenderCS	
Length Bias:	2	
This command is used to set various state variables in the VF stage.		
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: 0Ch 3DSTATE_VF Format: OpCode
	15:12	**Reserved** Project: All Format: MBZ
	11	**Reserved**
	10	**Reserved**
	9	**Reserved** Project: CHV, BSW Format: MBZ
	8	**Indexed Draw Cut Index Enable** Project: All Format: Enable If ENABLED, vertex indices in RANDOM 3DPRIMITIVE commands are compared to the Cut Index (specified below). When the vertex index matches the Cut Index any previous topology is terminated.If DISABLED, vertex indices are not compared to the Cut Index and are used strictly as indices into vertex buffers.This field can only be enabled for certain primitive topology types. Refer to the table later in this section for details.

3DSTATE_VF

	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Project: All Format: =n Total Length - 2
1	31:0	**Cut Index** Project: All This field specifies the index value that is considered the "cut index" which vertex indices are compared to if a Cut Index Enable is set. The Cut Index is compared to the fetched (and possibly-sign-extended) vertex index, and if these values are equal, the current primitive topology is terminated. Note that, for index buffers <32bpp, it is possible to set the Cut Index to a (large) value that will never match a sign-extended vertex index.

3DSTATE_VF_INSTANCING

3DSTATE_VF_INSTANCING

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command is used to control the "instancing" state associated with a specific vertex element.

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: 49h 3DSTATE_VF_INSTANCING Format: OpCode
	15:8	**Reserved**
	7:0	**DWord Length** Format: =n Total Length - 2 Value / Name: 1h: Excludes DWord (0,1) **[Default]** 43h: Context Restore
1	31:9	**Reserved**
	8	**Instancing Enable** Format: Enable Value / Name / Description: 0h / Disabled / This vertex element is not instanced and therefore vertices within instances can each receive different data for this vertex element. Within each instance, the source vertex data for this vertex element is determined according the the Vertex Access Type of the 3DPRIMITIVE command. Instance Data Step Rate is ignored for this vertex element.

<table>
<tr><th colspan="3">3DSTATE_VF_INSTANCING</th></tr>
<tr><td></td><td></td><td><table><tr><td>1h</td><td>Enabled</td><td>This vertex element is instanced and therefore vertices within instances will receive the same data for this vertex element. The source pointer for this particular vertex element will be (a) initialized at the start of 3DPRIMITIVE processing, (b) held constant for all vertices within an instance, and (c) advanced between instances as a function of Instance Data Step Rate.</td></tr></table></td></tr>
<tr><td></td><td>7:6</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>5:0</td><td>Vertex Element Index<table><tr><td>Format:</td><td>U6</td></tr></table>This field identifies which vertex element state is to be modified by this command.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,33]</td><td></td></tr></table></td></tr>
<tr><td>2</td><td>31:0</td><td>Instance Data Step Rate
If Instancing Enable is ENABLED, this field determines the rate at which data for this particular vertex element is changed between instances. Only after the number of instances specified by this field is generated is new (sequential) vertex element data provided. This process continues for each group of instances defined in the 3DPRIMTIVE command. For example, a value of 1 in this field causes new data to be supplied for this vertex element with each sequential (instance) group of vertices. A value of 2 causes every other instance group of vertices to be provided with new vertex element data. The special value of 0 causes all vertices of all instances generated by the 3DPRIMITIVE command to be provided with the same data for this vertex element. (The same effect can be achieved by setting this field to its maximum value.) If Instancing Enable is DISABLED, this field is ignored.</td></tr>
</table>

3DSTATE_VF_SGVS

3DSTATE_VF_SGVS

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command is used to control the insertion of the VertexID and InstanceID System-Generated Values (SGVs) into an input Vertex URB Entry (VUE) (available as input to a VS thread). VertexID and InstanceID insertion can be individually controlled. The insertion locations are specified as 128-bit element locations (starting at the beginning of the VUE) and the 32-bit component within those specified elements. The SGV values can be inserted either (a) within a valid vertex element (in which case the value overwrites the value specified via 3DSTATE_VERTEX_ELEMENTS) or (b) beyond the last valid vertex element written to the URB. This permits some orthogonality between the programming of vertex elements (which typically is known at draw time) and programming of SGV insertion (which is associated with the shader). There are some restrictions however (see below). If an SGV is inserted beyond the last valid vertex element, zeroes are first inserted in the VUE after the last valid vertex element up to and including the vertex element receiving an SGV. If both of the SGVs are enabled for insertion, the zeroes will extend to the last (largest index) vertex element receiving an SGV. Then the SGV(s) are inserted.

Programming Notes

Programming Restrictions:

- It is INVALID to store both the VertexID and InstanceID in the same element/component location within the VUE.
- The states programmed by this command overwrite the state programmed by any previous commands. I.e., VertexID and InstanceID (if enabled) can only be inserted in one component of a vertex.
- It is INVALID to insert an SGV value past the end of the VUE entry (as determined by VS URB Entry Allocation Size) or past the 33rd vertex element. Therefore the programming of VS URB Entry Allocation Size needs to comprehend any SGV insertion requirements.
- It is INVALID to use this command to overwrite any portion of a 64-bit vertex element component.
- It is INVALID to use this command to overwrite a EdgeFlag vertex element component or any vertex element beyond it.

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

3DSTATE_VF_SGVS

DWord	Bit	Description
	23:16	**3D Command Sub Opcode** Default Value: 4Ah 3DSTATE_VF_SGVS Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Project: All Format: =n Total Length - 2
1	31	**InstanceID Enable** Project: All Format: Boolean Value / Name / Description / Project: 0h / Disabled / InstanceID is not inserted / All 1h / Enabled / InstanceID is inserted / All
	30:29	**InstanceID Component Number** Project: All If InstanceID Enable is ENABLED, this field specifies the 32-bit component location (within the 4-component VUE) where it is inserted. If InstanceID Enable is DISABLED, this field is ignored. Value / Name / Description / Project: 0 / COMP_0 / If enabled, InstanceID is inserted in component 0 (.x) / All 1 / COMP_1 / If enabled, InstanceID is inserted in component 1 (.y) / All 2 / COMP_2 / If enabled, InstanceID is inserted in component 2 (.z) / All 3 / COMP_3 / If enabled, InstanceID is inserted in component 3 (.w) / All
	28:22	**Reserved**
	21:16	**InstanceID Element Offset** Project: All Format: U6 Offset of 128-bit element If InstanceID Enable is ENABLED, this field specifies the VUE element offset of the 128-bit element where it is to be inserted. The InstanceID Component Number specifies where in the specified element it is inserted. Value / Name: [0,33] /

3DSTATE_VF_SGVS

	15	**VertexID Enable** Project: All Format: Boolean

Value	Name	Description	Project
0h	Disabled	VertexID is not inserted	All
1h	Enabled	VertexID is inserted	All

	14:13	**VertexID Component Number** Project: All If VertexID Enable is ENABLED, this field specifies the 32-bit component location (within the 4-component VUE) where it is inserted.If VertexID Enable is DISABLED, this field is ignored.

Value	Name	Description	Project
0	COMP_0	If enabled, VertexID is inserted in component 0 (.x)	All
1	COMP_1	If enabled, VertexID is inserted in component 1 (.y)	All
2	COMP_2	If enabled, VertexID is inserted in component 2 (.z)	All
3	COMP_3	If enabled, VertexID is inserted in component 3 (.w)	All

	12:6	**Reserved** Project: All Format: MBZ
	5:0	**VertexID Element Offset** Project: All Format: U6 Offset of 128-bit element If VertexID Enable is ENABLED, this field specifies the VUE element offset of the 128-bit element where it is to be inserted. The VertexID Component Number specifies where in the specified element it is inserted. This is also the vertex element index. If VertexID Enable is DISABLED, this field is ignored.

Value	Name
[0,33]	

3DSTATE_VF_STATISTICS

<table>
<tr><th colspan="3">3DSTATE_VF_STATISTICS</th></tr>
<tr><td colspan="3">Project: CHV, BSW
Source: RenderCS
Length Bias: 1</td></tr>
<tr><td colspan="3">The VF stage tracks two pipeline statistics, the number of vertices fetched and the number of objects generated. VF will increment the appropriate counter for each when statistics gathering is enabled by issuing the 3DSTATE_VF_STATISTICS command with the [Statistics Enable] bit set.</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="6">0</td><td>31:29</td><td>Command Type<table><tr><td>Default Value:</td><td>3h GFXPIPE</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td>28:27</td><td>Command SubType<table><tr><td>Format:</td><td>Opcode</td></tr></table><table><tr><th>Value</th><th>Name</th><th>Project</th></tr><tr><td>1h</td><td>Pipelined, Single DWord [Default]</td><td>CHV, BSW</td></tr></table></td></tr>
<tr><td>26:24</td><td>3D Command Opcode<table><tr><td>Default Value:</td><td>0h 3DSTATE_PIPELINED</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table>GFXPIPE[28:27 = 1h, 26:24 = 0h, 23:16 = 0Bh] (Pipelined, Single DWord)</td></tr>
<tr><td>23:16</td><td>3D Command Sub Opcode<table><tr><td>Default Value:</td><td>0Bh 3DSTATE_VF_STATISTICS</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table>GFXPIPE[28:27 = 1h, 26:24 = 0h, 23:16 = 0Bh] (Pipelined, Single DWord)</td></tr>
<tr><td>15:1</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>0</td><td>Statistics Enable<table><tr><td>Format:</td><td>Enable</td></tr></table>If ENABLED, VF will increment the pipeline statistics counters IA_VERTICES_COUNT and IA_PRIMITIVES_COUNT for each vertex fetched and each object output, respectively, for 3DPRIMITIVE commands issued subsequently. If DISABLED, these counters will not be incremented for subsequent 3DPRIMITIVE commands.</td></tr>
</table>

3DSTATE_VF_TOPOLOGY

3DSTATE_VF_TOPOLOGY

Project: CHV, BSW
Source: RenderCS
Length Bias: 2

This command specifies the VF stage's Topology state which can be used to override the Primitive Topology Type in subsequent 3DPRIMITIVE commands.

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: 4Bh 3DSTATE_VF_TOPOLOGY Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Project: All Format: =n Total Length - 2
1	31:6	**Reserved** Project: All Format: MBZ
	5:0	**Primitive Topology Type** Project: All Format: 3D_Prim_Topo_Type [CHV, BSW] This field specifies the VF stage's Topology state.

3DSTATE_VIEWPORT_STATE_POINTERS_CC

3DSTATE_VIEWPORT_STATE_POINTERS_CC

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The 3DSTATE_VIEWPORT_STATE_POINTERS_CC command is used to define the location of fixed functions' viewport state table.

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: 23h 3DSTATE_VIEWPORT_STATE_POINTERS Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:5	**CC Viewport Pointer** Project: All Format: DynamicStateOffset[31:5]CC_VIEWPORT*16 Specifies the 32-byte aligned address offset of the CC_VIEWPORT state. This offset is relative to the Dynamic State Base Address.
	4:0	**Reserved** Project: All Format: MBZ

3DSTATE_VIEWPORT_STATE_POINTERS_SF_CLIP

3DSTATE_VIEWPORT_STATE_POINTERS_SF_CLIP

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The 3DSTATE_VIEWPORT_STATE_POINTERS_CLIP command is used to define the location of fixed functions' viewport state table.

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: 21h 3DSTATE_VIEWPORT_STATE_POINTERS_SF_CLIP Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:6	**SF Clip Viewport Pointer** Project: All Format: DynamicStateOffset[31:6]SF_CLIP_VIEWPORT*16 Specifies the 64-byte aligned address offset of the SF_CLIP_VIEWPORT state. This offset is relative to the Dynamic State Base Address.
	5:0	**Reserved** Project: Format: MBZ

3DSTATE_VS

3DSTATE_VS

Project:	CHV, BSW
Source:	RenderCS, PositionCS
Length Bias:	2

Description	Project
This command specifies most of the state used by the Vertex Shader (VS) stage.	CHV, BSW

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: 10h 3DSTATE_VS Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 7h Excludes DWord (0,1) Project: All Format: =n Total Length - 2
1..2	63:6	**Kernel Start Pointer** Project: All Format: InstructionBaseOffset[63:6]Kernel This field specifies the starting location of the kernel program run by threads spawned by the VS pipeline stage. It is specified as a 64-byte-granular offset from the Instruction Base Address. This field is ignored if VS Function Enable is DISABLED.
	5:0	**Reserved**

3DSTATE_VS

DWord 3, Bit 31 — Single Vertex Dispatch

Project:	
Format:	U1 Enumerated Type

When this bit is set, SIMD4x2 VS threads will only process a single vertex, otherwise SIMD4x2 threads will process either one or two vertices. This field is ignored if **SIMD8 Dispatch Enable** is set.

Value	Name	Description	Project
0h	Multiple	Dual vertex SIMD4x2 thread dispatches are allowed.	All
1h	Single	Single vertex SIMD4x2 thread dispatches are forced.	All

Bit 30 — Vector Mask Enable

Project:	All

Upon subsequent VS 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	Project
0h	Dmask	The EU will use the Dispatch Mask (supplied by the VS stage) for instruction execution.	All
1h	Vmask	The EU will use the Vector Mask (derived from the Dispatch Mask) for instruction execution.	All

Programming Notes
Under normal conditions SW shall specify DMask, as the VS stage will provide a Dispatch Mask appropriate to SIMD4x2 or SIMD8 thread execution (as a function of SIMD8 Dispatch Enable). E.g., for SIMD4x2 thread execution, the VS 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).

Bits 29:27 — Sampler Count

Project:	All
Format:	U3

This field specifies (in multiples of 4) the number of sets of sampler state that will be prefetched for use by the VS kernel. While the prefetching of sampler state is optional and does not impact functionality, it may improve performance.
This field is ignored if the Function Enable state is set to DISABLED.

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

Bit 26 — Reserved

3DSTATE_VS

Project:	All
Format:	MBZ

25:18 Binding Table Entry Count

Project:	All
Format:	U8

Description
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 VS Function Enable is DISABLED.
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	Project
When HW binding table bit is set, it is assumed that the Binding Table Entry Count field will be generated at JIT time.	CHV, BSW

17 Thread Dispatch Priority

Project:	All
Format:	U1 Enumerated Type

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

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

16 Floating Point Mode

Project:	All
Format:	U1 Enumerated Type

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

Value	Name	Description	Project
0h	IEEE-754	Use IEEE-754 Rules	All
1h	Alternate	Use Alternate Rules	All

3DSTATE_VS

	15:14	**Reserved** Project: All Format: MBZ
	13	**Illegal Opcode Exception Enable** Project: All 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 VS Function Enable is DISABLED.
	12	**Accesses UAV** Format: Enable This field must be set when VS has a UAV access. **Programming Notes** / **Project** This field must not be set when VS Function Enable is disabled. Workaround: If the vertex shader is the last shader to have UAV access, a PIPE_CONTROL with **CS_STALL** must be sent before the 3dprimitive using the UAV access. — CHV, BSW
	11:8	**Reserved**
	7	**Software Exception Enable** Project: All 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 VS Function Enable is DISABLED.
	6:0	**Reserved** Project: All Format: MBZ
4..5	63:10	**Scratch Space Base Pointer** Project: All 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. In 64b OS all pointers need to be seen by SW as 48b. HW does not support a Scratch Space Base Pointer larger than 32b, therefore SW must ensure the scratch space base offset upper is set to 0's. But the DWORD must be in the command for SW to right the pointer to.

3DSTATE_VS

DWord	Bit	Description
	9:4	**Reserved** Project: All Format: MBZ
	3:0	**Per-Thread Scratch Space** Project: All 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 VS Function Enable is DISABLED. Value: [0,11] — Name: — Description: 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
	24:20	**Dispatch GRF Start Register For URB Data** Project: All Format: U5 Specifies the starting GRF number for the URB portion (URB constants and vertices) of the thread payload. This field is ignored if VS Function Enable is DISABLED. Value: [0,31] — Name: — Description: indicating GRF [R0, R31]
	19:17	**Reserved** Project: All Format: MBZ
	16:11	**Vertex URB Entry Read Length** Format: U6 Specifies the number of pairs of 128-bit vertex elements to be passed into the payload for each vertex. This field is ignored if VS Function Enable is DISABLED. For SIMD4x2 dispatch, each vertex element requires one GRF of payload data, therefore the number of GRFs with vertex data will be double the value programmed in this field. For SIMD8 dispatch, each vertex element requires 4 GRFs of payload data, therefore the number of GRFs with vertex data will be 8 times the value programmed in this field. The EU limit of 128 GRFs imposes a maximum limit of 30 elements per vertex pushed into the payload, though the practical limit may be lower. If input vertices exceed

3DSTATE_VS

DWord	Bit	Description
		the practical limit, software must decide between resorting to pulling elements during thread execution or dropping back to SIMD4x2 dispatch. Note that the VUE is used for both input and output, so when using the pull-model software must ensure inputs are not overwritten before last use.

Value	Name	Description
[1,63]		if SIMD8 dispatch disabled
[0,15]		if SIMD8 dispatch enabled

10 — Reserved

Project:	All
Format:	MBZ

9:4 — Vertex URB Entry Read Offset

Project:	All
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 VS Function Enable is DISABLED.

Value	Name
[0,63]	

3:0 — Reserved

Project:	All
Format:	MBZ

DWord 7, 31:23 — Maximum Number of Threads

Project:	CHV, BSW
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. This field is ignored if VS Function Enable is DISABLED.

Value	Name	Description	Project
[0,503]		indicating thread count of [1,504]	CHV, BSW
[0,79]		indicating thread count of [1,80]	CHV, BSW

22 — Reserved

Project:	CHV, BSW
Format:	MBZ

21:11 — Reserved

3DSTATE_VS

	10	**Statistics Enable** Project: All Format: Enable If ENABLED, the VS stage will perform statistics gathering. See the Statistics Gathering subsection. If DISABLED, statistics information associated with the VS stage will be left unchanged.
	9	**Reserved** Project: CHV, BSW Format: MBZ
	8:3	**Reserved** Project: All Format: MBZ
	2	**SIMD8 Dispatch Enable** Project: All Format: Enable This field determines how VS threads are dispatched and how the thread payloads are generated. The setting of this field must agree with how the VS kernel was compiled. If ENABLED, SIMD8 VS thread dispatches are performed. The **Single Vertex Dispatch** field is ignored. If DISABLED, SIMD4x2 thread dispatches are performed. The **Single Vertex Dispatch** field can be used to force single-vertex dispatches.
	1	**Vertex Cache Disable** Project: All Format: Disable This bit controls the operation of the Vertex Cache. This field is always used. If the Vertex Cache is DISABLED and the VS Function is ENABLED, the Vertex Cache is not used and all incoming vertices will be passed to VS threads. If the Vertex Cache is ENABLED and the VS Function is ENABLED, only incoming vertices that do not hit in the Vertex Cache will be passed to VS threads. If the Vertex Cache is ENABLED and the VS Function is DISABLED, input vertices that miss in the Vertex Cache will be assembled and written to the URB (by the VF stage), and subsequently passed through the VS stage unmodified (i.e, no VS threads are spawned). The Vertex Cache is invalidated whenever the Vertex Cache becomes DISABLED, whenever the VS Function Enable toggles, between 3DPRIMITIVE commands and between instances within a 3DPRIMITIVE command. **Programming Notes** See the Vertex Caching subsection for details on implicit Vertex Cache disabling and the "chicken bit" available to turn off the implicit disable.

3DSTATE_VS

DWord	Bit	Description
	0	**Function Enable** Format: Enable This bit determines whether or not the VS stage spawns VS threads, which comprises the bulk of the VS stage functionality. If ENABLED, VS threads may be spawned to process VF-generated vertices before the resulting vertices are passed down the pipeline. If DISABLED, VF-generated vertices will pass thru the VS function and are sent down the pipeline unmodified. The Vertex Cache (if enabled) is still available.
8	31:28	**Reserved** Project: All Format: MBZ
	27	**Reserved** Project: Format: MBZ
	26:21	**Vertex URB Entry Output Read Offset** Project: All Format: U6 Specifies the offset (in 256-bit units) at which Vertex URB data is to be read from the URB by the Setup Back-End (SBE) function. The offset programmed will specify the start of Attribute 0 to be passed in subsequent Pixel Shader thread payloads. Refer to the Attribute Interpolator Setup documentation. Value: [0,63] — Name: (none) **Programming Notes** As the vertex header data located at the start of the Vertex URB entry is typically only used by 3D pipeline FFs (i.e., Clipper, Setup FrontEnd) and not required as interpolated attributes in PIxel Shader threads, it is expected that SW will program this Start Offset skip over the vertex header. This offset value is ignored if SBE's Number of SF Attributes state is programmed to 0 (i.e., no attributes are defined beyond the position read from the Vertex Header)
	20:16	**Vertex URB Entry Output Length** Project: All Format: U5 Specifies the amount of Vertex Attribute URB data to be read by the Setup Back-End function for each Vertex URB entry, in 256-bit units. The attribute data will be read starting at the offset specified by the Vertex URB Entry Output Read Offset state. Value: [1,16] — Name: (none)

3DSTATE_VS

		Programming Notes This length value is ignored if SBE's Number of SF Attributes state is programmed to 0 (i.e., no attributes are defined beyond the position read from the Vertex Header).
	15:8	**User Clip Distance Clip Test Enable Bitmask** Project: All Format: mask[8] This 8 bit mask field selects which of the 8 Clip Distance Values (if any) are to be included in the Clip stage's trivial reject / trivial accept / must clip determination function. The ClipDistance Values (if present) are located in DW8-15 of the VUE Vertex Header located at the beginning of VUE URB entries. Bit 0 of this field corresponds to Clip Distance Value 0.
	7:0	**User Clip Distance Cull Test Enable Bitmask** Project: All Format: mask[8] This 8 bit mask field selects which of the 8 Clip Distance Values (if any) are to be included in the Clip stage's trivial reject / trivial accept determination function. Note that must clip determination is not included in this function. The ClipDistance Values (if present) are located in DW8-15 of the VUE Vertex Header located at the beginning of VUE URB entries. Bit 0 of this field corresponds to Clip Distance Value 0.

3DSTATE_WM_CHROMAKEY

3DSTATE_WM_CHROMAKEY		
Project: CHV, BSW		
Source: RenderCS		
Length Bias: 2		
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: 4Ch 3DSTATE_WM__CHROMAKEY Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**Dword Length** Default Value: 0h Excludes Dword (0,1) Project: All Format: =n Total Length - 2
1	31	**ChromaKey Kill Enable** Project: All Format: Enable If ENABLED, indicates that at least one of the attached samplers has ChromaKeyKill enabled.
	30:0	**Reserved** Project: All Format: MBZ

3DSTATE_WM_DEPTH_STENCIL

3DSTATE_WM_DEPTH_STENCIL

Project: CHV, BSW
Source: RenderCS
Length Bias: 2

This command replaces the indirect state DEPTH_STENCIL_STATE with an inline state command.

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: 4Eh 3DSTATE_WM_DEPTH_STENCIL Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**Dword Length** Project: All Format: =n Total Length - 2 Value: 01h — Name: Excludes Dword (0,1) **[Default]** — Project: CHV, BSW
1	31:29	**Stencil Fail Op** Project: All Format: 3D_Stencil_Operation [CHV, BSW] This field specifies the operation to perform on the Stencil Buffer when the (front face) stencil test fails. **Programming Notes** if all three stencil ops (Stencil Fail, Stencil Pass Depth Fail, and Stencil Pass Depth Pass) are KEEP, ZERO, or REPLACE, the stencil buffer is not read.

3DSTATE_WM_DEPTH_STENCIL

	28:26	**Stencil Pass Depth Fail Op** Project: All Format: 3D_Stencil_Operation [CHV, BSW] This field specifies the operation to perform on the Stencil Buffer when the (front face) stencil test passes but the depth pass fails.
	25:23	**Stencil Pass Depth Pass Op** Project: All Format: 3D_Stencil_Operation [CHV, BSW] This field specifies the operation to perform on the Stencil Buffer when the (front face) stencil test passes but the depth test passes.
	22:20	**Backface Stencil Test Function** Project: All Format: 3D_Compare_Function
	19:17	**Backface Stencil Fail Op** Project: All Format: 3D_Stencil_Operation [CHV, BSW]
	16:14	**Backface Stencil Pass Depth Fail Op** Project: All Format: 3D_Stencil_Operation [CHV, BSW] This field specifies the operation to perform on the Stencil Buffer when the stencil test passes but the depth pass fails.
	13:11	**Backface Stencil Pass Depth Pass Op** Project: All Format: 3D_Stencil_Operation [CHV, BSW] This field specifies the operation to perform on the Stencil Buffer when the stencil test passes and the depth pass passes (or is disabled).
	10:8	**Stencil Test Function** Project: All Format: 3D_Compare_Function This field specifies the comparison function used in the (front face) StencilTest function.
	7:5	**Depth Test Function** Project: All Format: 3D_Compare_Function Specifies the comparison function used in DepthTest function.

3DSTATE_WM_DEPTH_STENCIL

		Programming Notes If the Depth Test Function is ALWAYS or NEVER, the depth buffer is not read.
	4	**Double Sided Stencil Enable** Project: All Format: Enable Enable doubled sided stencil operations.
	3	**Stencil Test Enable** Project: All Format: Enable Enables StencilTest function of the Pixel Processing pipeline.
	2	**Stencil Buffer Write Enable** Project: All Format: Enable Enables writes to the Stencil Buffer.
	1	**Depth Test Enable** Project: All Format: Enable Enables the DepthTest function of the Pixel Processing pipeline.

Programming Notes

If the Depth Test Function is ALWAYS or NEVER, the depth buffer is not read.

4 **Double Sided Stencil Enable**

Project:	All
Format:	Enable

Enable doubled sided stencil operations.

Value	Name	Description	Project
0h	False	Double Sided Stencil Disabled	All
1h	True	Double Sided Stencil Enabled	All

Programming Notes

- Back-facing primitives have a vertex winding order opposite to the currently selected Front Winding state.
- Culling of primitives is not affected by the double sided stencil state
- Back-facing primitives will be rendered, honoring all current device state, as though it were a front-facing primitive with no implicitly overloaded state.

3 **Stencil Test Enable**

Project:	All
Format:	Enable

Enables StencilTest function of the Pixel Processing pipeline.

Programming Notes

If any of the render targets are YUV format, this field must be disabled.

2 **Stencil Buffer Write Enable**

Project:	All
Format:	Enable

Enables writes to the Stencil Buffer.

Programming Notes

If this field is enabled, Stencil Test Enable must also be enabled.

1 **Depth Test Enable**

Project:	All
Format:	Enable

Enables the DepthTest function of the Pixel Processing pipeline.

Value	Name	Project
0h	Disable	All
1h	Enable	All

Programming Notes

3DSTATE_WM_DEPTH_STENCIL

		If any of the render targets are YUV format, this field must be disabled.
	0	**Depth Buffer Write Enable** Project: All Format: Enable Enables writes to the Depth Buffer. **Programming Notes** / **Project** A Depth Buffer must be defined before enabling writes to it, or operation is UNDEFINED. This bit must not be set when WM_INT::RT Independent Rasterization Enable is true. — CHV, BSW **Workaround** / **Project** If Depth_Test_Enable = 1 AND Depth_Test_func = EQUAL, the Depth_Write_Enable must be set to 0 — CHV, BSW
2	31:24	**Stencil Test Mask** Project: All Format: U8 This field specifies a bit mask applied to stencil test values. Both the stencil reference value and value read from the stencil buffer will be logically ANDed with this mask before the stencil comparison test is performed.
	23:16	**Stencil Write Mask** Project: All Format: U8 This field specifies a bit mask applied to stencil buffer writes. Only those stencil buffer bits corresponding to bits set in this mask will be modified.
	15:8	**Backface Stencil Test Mask** Project: All Format: U8 This field specifies a bit mask applied to backface stencil test values. Both the stencil reference value and value read from the stencil buffer will be logically ANDed with this mask before the stencil comparison test is performed.
	7:0	**Backface Stencil Write Mask** Project: All Format: U8 This field specifies a bit mask applied to backface stencil buffer writes. Only those stencil buffer bits corresponding to bits set in this mask will be modified.

3DSTATE_WM

3DSTATE_WM

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

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: 14h 3DSTATE_WM Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Project: All Format: =n Total Length - 2
1	31	**Statistics Enable** Project: All Format: Enable If ENABLED, the Windower and pixel pipeline will engage in statistics gathering. If DISABLED, statistics information associated with this FF stage will be left unchanged. See Statistics Gathering. **Programming Notes** This bit must be disabled if any of these bits is set: 3DSTATE_WM::**Legacy Depth Buffer Clear**, 3DSTATE_WM::**Legacy Hierarchical Depth Buffer Resolve Enable** or 3DSTATE_WM::**Legacy Depth Buffer Resolve Enable**.

3DSTATE_WM

	30	**Legacy Depth Buffer Clear Enable**
		Project: All
		Format: Enable
		When set, the depth buffer is initialized as a side-effect of rendering pixels.
		Programming Notes
		If this field is enabled,
		1. the **Depth Test Enable** field in DEPTH_STENCIL_STATE must be disabled.
		2. 3DSTATE_DEPTH_BUFFER::Depth Write Enable must be set.
		3. 3DSTATE_DEPTH_BUFFER::Stencil Write Enable must be set if 3DSTATE_STENCIL_BUFFER::Stencil buffer enable is set. Additionally the following must be set to the correct values.
		    1. DEPTH_STENCIL_STATE::Stencil Write Mask must be 0xFF
		    2. DEPTH_STENCIL_STATE::Stencil Test Mask must be 0xFF
		    3. DEPTH_STENCIL_STATE::Back Face Stencil Write Mask must be 0xFF
		    4. DEPTH_STENCIL_STATE::Back Face Stencil Test Mask must be 0xFF
		Refer to section 0 "Depth Buffer Clear" for additional restrictions when this field is enabled. If this field is enabled,**Pixel Shader Kill Pixel** must be disabled.

	29	**Reserved** Project: All Format: MBZ
	28	**Legacy Depth Buffer Resolve Enable** Project: All Format: Enable When set, the depth buffer is made to be consistent with the hierarchical depth buffer as a side-effect of rendering pixels. This is intended to be used when the depth buffer is to be used as a surface outside of the 3D rendering operation. **Programming Notes** If this field is enabled, 1. the **Legacy Depth Buffer Clear** and **Legacy Hierarchical Depth Buffer Resolve Enable** fields must both be disabled. 2. 3DSTATE_DEPTH_BUFFER::Depth Write Enable must be set. Refer to section 11.5.4.2 "Depth Buffer Resolve" for additional restrictions when this field is enabled. **If Hierarchical Depth Buffer Enable** is disabled, enabling this field will have no effect.
	27	**Legacy Hierarchical Depth Buffer Resolve Enable** Project: All Format: Enable When set, the hierarchical depth buffer is made to be consistent with the depth buffer as a side-effect of rendering pixels. This is intended to be used when the depth buffer has been modified

3DSTATE_WM

outside of the 3D rendering operation.

Programming Notes

If this field is enabled,

1. the **Legacy Depth Buffer Clear** and **Legacy Depth Buffer Resolve Enable** fields must both be disabled.
2. 3DSTATE_DEPTH_BUFFER::Depth Write Enable must be set.

Refer to section 11.5.4.3 "Hierarchical Depth Buffer Resolve" for additional restrictions when this field is enabled. If **Hierarchical Depth Buffer Enable** is disabled, enabling this field will have no effect. **Performance Note:** expect the hierarchical depth buffer's impact on performance to be reduced for some period of time after this operation is performed, as the hierarchical depth buffer is initialized to a state that makes it ineffective. Further rendering will tend to bring the hierarchical depth buffer back to a more effective state.

26 **Legacy Diamond Line Rasterization**

Project:	All
Format:	Enable

This bit, if ENABLED, indicates that the Windower will rasterize zero width lines using the DX9 rasterization rules. If DISABLED, the Windower will rasterize zero width lines using the DX10 rasterization rules (see Strips Fans chapter).

25:23 **Reserved**

Project:	All
Format:	MBZ

22:21 **Early Depth/Stencil Control**

Project:	CHV, BSW
Format:	U2 Enumerated Type

This field specifies the behavior of early depth/stencil test.

Value	Name	Description	Project
0h	NORMAL	Depth/Stencil Test/Write behaves as if it happens post-shader, however the pixel shader is not necessarily executed if the pixel fails depth or stencil test (this is the legacy behavior)	All
1h	PSEXEC	Depth/Stencil Test/Write behaves as if it happens post-shader, and the pixel shader is executed if the pixel fails depth or stencil test (although pre-shader actions such as primitive inclusion, stipple, etc. will still cause the shader not to execute)	All
2h	PREPS	Depth/Stencil Test/Write behaves as if it happens pre-shader. The pixel shader is not executed if the pixel fails depth or stencil test. Depth and stencil writes occur even if the pixel is killed by the shader or post-shader by alpha test, etc. Depth output by the pixel shader is ignored.	All
3h	Reserved		All

3DSTATE_WM

		Programming Notes The Early Depth/Stencil Control field cannot be set to PREPS (value = 2h) if ForceKillpix = ForceON or Forced Thread Dispatch = ForceON
	20:19	**Force Thread Dispatch Enable**

Project:	All

Value	Name	Description	Project
0h	Normal	WM_INT::ThreadDispatchEnable is computed normally	All
1h	ForceOff	Forces WM_INT::ThreadDispatchEnable Off	All
2h	ForceON	Forces WM_INT::ThreadDispatchEnable On	All
3h	Reserved		All

Programming Notes
This should must always be set to Normal, except for driver debug. This field should not be tested for functional validation

18:17 **Position ZW Interpolation Mode**

Project:	All
Format:	U2 Enumerated Type

This field elects "interpolation mode" associated with the Position Z (source depth) and W coordinates passed in the PS payload when the PS requires Position as input. This field does not determine whether these coordinates are actually included in the payload (see Pixel Shader Requires Depth, Pixel Shader Requires W).

Value	Name	Description	Project
0h	INTERP_PIXEL	Evaluate Z & W at the pixel center or UL corner (as specified by Pixel Location of 3DSTATE_MULTISAMPLE)	All
1h	Reserved		All
2h	INTERP_CENTROID		All
3h	INTERP_SAMPLE		All

Programming Notes
WM_INT::RT Independent Rasterization Enable must be disabled in order to select INTERP_SAMPLE.
MSDISPMODE_PERSAMPLE is required in order to select INTERP_SAMPLE.

16:11 **Barycentric Interpolation Mode**

Project:	All
Format:	Enable[6]

Controls which barycentric interpolation terms must be passed into the pixel shader kernel. Bit 0: Perspective Pixel Location barycentric is required Bit 1: Perspective Centroid barycentric is

3DSTATE_WM

		required Bit 2: Perspective Sample barycentric is required Bit 3: Non-perspective Pixel Location barycentric is required Bit 4: Non-perspective Centroid barycentric is required Bit 5: Non-perspective Sample barycentric is required **Programming Notes** If contiguous dispatch modes are enabled, only bit 3 (non-perspective pixel location) can be set, all other bits in this field must be zero. Pixel Location below refers to either the upper left corner or pixel center depending on the **Pixel Location** state of 3DSTATE_MULTISAMPLING). MSDISPMODE_PERSAMPLE is required in order to select Perspective Sample or Non-perspective Sample barycentric coordinates.
	10	**Reserved** Project: All Format: MBZ
	9:8	**Line End Cap Antialiasing Region Width** Project: All Format: U2 This field specifies the distances over which the coverage of anti-aliased line end caps are computed.

Value	Name	Description	Project
0h	0.5 pixels	0.5 pixels	All
1h	1.0 pixels	1.0 pixels	All
2h	2.0 pixels	2.0 pixels	All
3h	4.0 pixels	4.0 pixels	All

	7:6	**Line Antialiasing Region Width** Project: All Format: U2 This field specifies the distance over which the anti-aliased line coverage is computed.

Value	Name	Description	Project
0h	0.5 pixels	0.5 pixels	All
1h	1.0 pixels	1.0 pixels	All
2h	2.0 pixels	2.0 pixels	All
3h	4.0 pixels	4.0 pixels	All

	5	**Reserved** Project: All Format: MBZ
	4	**Polygon Stipple Enable** Project: All Format: Enable Enables the Polygon Stipple function.

3DSTATE_WM

	3	**Line Stipple Enable**

Project:	All
Format:	Enable

Enables the Line Stipple function.

	2	**Point Rasterization Rule**

Project:	All
Format:	3D_RasterizationRule

This field specifies the rasterization rules to be applied whenever the edges of a point primitive fall exactly on a pixel sampling point.

Value	Name	Description	Project
0h	RASTRULE_UPPER_LEFT	To match "normal" upper left rules for surface primitives	All
1h	RASTRULE_UPPER_RIGHT	To match OpenGL point rasterization rules (round to + infinity, where this is the upper right direction wrt OpenGL screen origin of lower left).	All

	1:0	**Force Kill Pixel Enable**

Project:	All

Value	Name	Description	Project
0h	Normal	WM_INT:: Pixel Shader Kill Pixel is computed normally	All
1h	ForceOff	Forces WM_INT:: Pixel Shader Kill Pixel Off	All
2h	ForceON	Forces WM_INT:: Pixel Shader Kill Pixel On	All
3h	Reserved		All

Programming Notes
This should must always be set to Normal, except for driver debug. This field should not be tested for functional validation

3DSTATE_WM_HZ_OP

3DSTATE_WM_HZ_OP

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

This command provides for clearing Z and/or stencil or resolving either HZ buffer or Z buffer.

Programming Notes
3DSTATE_MULTISAMPLE packet must be used prior to this packet to change the Number of Multisamples. This packet must not be used to change Number of Multisamples in a rendering sequence. 3DSTATE_RASTER if used must be programmed prior to using this packet.
This command does support predication from the use of the MI_PREDICATE register. To predicate depth clears and resolves on [CHV, BSW] you must fall back to using the 3D_PRIMITIVE or GPGPU_WALKER commands.
As this command generates an implicit rectangle, SW must make sure any MMIO register writes following WM_HZ_OP must be preceded by **PIPE_CONTROL** with **Command Streamer Stall Enable** bit set.

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: 52h 3DSTATE_WM_HZ_OP Format: OpCode
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**Dword Length** Default Value: 03h Excludes Dword (0,1) Project: All Format: =n

3DSTATE_WM_HZ_OP

		Total Length - 2
1	31	**Stencil Buffer Clear Enable** Project: All Format: Enable When set, the stencil buffer is initialized. **Programming Notes** If this field is enabled, 1. the **Depth Buffer Resolve Enable** and **Hierarchical Depth Buffer Resolve Enable** fields must both be disabled. 2. 3DSTATE_DEPTH_BUFFER::Stencil Write Enable must be set if 3DSTATE_STENCIL_BUFFER::Stencil buffer enable is set.
	30	**Depth Buffer Clear Enable** Project: All Format: Enable When set, the depth buffer is initialized. **Programming Notes** If this field is enabled, 1. the **Depth Buffer Resolve Enable** and **Hierarchical Depth Buffer Resolve Enable** fields must both be disabled. 2. 3DSTATE_DEPTH_BUFFER::Depth Write Enable must be set.
	29	**Scissor Rectangle Enable** Project: All Format: Enable Enables operation of Scissor Rectangle. **Programming Notes** In order get the functionality right if this bit is disabled, driver must clip the clear rectangle to scissor rectangle if scissor test is enabled before clearing. **Workaround** / **Project** Workaround: Due to a Hardware issue this bit must not be set. / CHV, BSW
	28	**Depth Buffer Resolve Enable** Project: All Format: Enable When set, the depth buffer is made to be consistent with the hierarchical depth buffer as a side-effect of rendering pixels. This is

3DSTATE_WM_HZ_OP

<table>
<tr><td></td><td></td><td>intended to be used when the depth buffer is to be used as a surface outside of the 3D rendering operation.
Programming Notes
If this field is enabled,
1. the Depth Buffer Clear and Hierarchical Depth Buffer Resolve Enable fields must both be disabled.
2. 3DSTATE_DEPTH_BUFFER::Depth Write Enable must be set.
Refer to section 11.5.4.2 "Depth Buffer Resolve" for additional restrictions when this field is enabled. If Hierarchical Depth Buffer Enable is disabled, enabling this field will have no effect.</td></tr>
<tr><td></td><td>27</td><td>Hierarchical Depth Buffer Resolve Enable
Project: All
Format: Enable
When set, the hierarchical depth buffer is made to be consistent with the depth buffer as a side-effect of rendering pixels. This is intended to be used when the depth buffer has been modified outside of the 3D rendering operation.
Programming Notes
If this field is enabled,
1. the Depth Buffer Clear and Depth Buffer Resolve Enable fields must both be disabled.
2. 3DSTATE_DEPTH_BUFFER::Depth Write Enable must be set.
Refer to section 11.5.4.3 "Hierarchical Depth Buffer Resolve" for additional restrictions when this field is enabled. If Hierarchical Depth Buffer Enable is disabled, enabling this field will have no effect. Performance Note: expect the hierarchical depth buffer's impact on performance to be reduced for some period of time after this operation is performed, as the hierarchical depth buffer is initialized to a state that makes it ineffective. Further rendering will tend to bring the hierarchical depth buffer back to a more effective state.</td></tr>
<tr><td></td><td>26</td><td>Pixel Position Offset Enable
Project: All
Format: U1 Enumerated Type
Enables the device to offset pixel positions by 0.5 both in horizontal and vertical directions.
Programming Notes
Setting this field along with setting the Pixel Location to upper left and number of multisamples to greater than one will cause the device to offset pixel postions by 0.5 both in horizontal and vertical</td></tr>
</table>

3DSTATE_WM_HZ_OP

		directions.It is to be noted this is done to adjust the pixel co-ordinate system to DX9 like, so any WM_HZ_OP screen space rectangles (eg: legacy HiZ Clear, Resolve etc) generated internally by driver in this mode needs to be aware of this offset adjustment and send the rectangles according to alignment restriction taking this offset adjustment into consideration. SW can choose to set this bit only for DX9 API. DX10/OGL API's should not have any effect by setting or not setting this bit.
	25	**Full Surface Depth Clear**

Project:	All
Format:	Enable

Programming Notes
Setting this field to "1" along with "Depth buffer clear" will cause all the pixels/samples in an 8x4 block in the HIZ buffer to be cleared. If "Stc-buffer clear" is also set, then all pixels/samples in a 8x8 block of STC buffer will be cleared to the stc-ref value. Software must set this only when the APP requires the entire Depth surface to be cleared. Setting this field to "1" for STC-buffer only clear without "depth buffer clear" will cause all the pixels/samples in an 8x8 block in the STC buffer to get the stc-ref value.

24 **Reserved**

Project:	All
Format:	MBZ

23:16 **Stencil Clear Value**

Format:	U8.0

This field specifies the stencil clear value.

15:13 **Number of Multisamples**

Project:	All
Format:	U3 Enumerated Type

This field specifies how many samples/pixel exist in the Depth Buffer and Stencil buffers, as log2(#samples).

Value	Name	Description	Project
0h	1	1 sample/pixel	All
1h	2	2 samples/pixel	All
2h	4	4 samples/pixel	All
3h	8	8 samples/pixel	All
4h	Reserved		CHV, BSW
5h-7h	Reserved		All

3DSTATE_WM_HZ_OP

	12:0	**Reserved** Project: All
2 **Programming Notes:** The clear rectangle x and y min and max values must be shifted by the LOD level; i.e. the hardware does not include the LOD in this function. Hence to clear any particular X, Y from the base level, to clear the contents at level "LOD" use (X»LOD) and (Y»LOD).	31:16	**Clear Rectangle Y Min** Project: All Format: U16 in Pixels from Depth Buffer origin (upper left corner) Specifies Ymin value of (inclusive) of clear rectangle with the Depth Buffer, used for clipping. Pixels with Y coordinates less than Ymin will not be affected. **Value** / **Name** / **Project** [0,16383] / (Device ignores bits 31:30) / CHV, BSW
	15:0	**Clear Rectangle X Min** Project: All Format: U16 in Pixels from Depth Buffer origin (upper left corner) Specifies Xmin value of (inclusive) of clear rectangle with the Depth Buffer, used for clipping. Pixels with X coordinates less than or equal to Xmin will not be affected. **Value** / **Name** / **Project** [0,16383] / (Device ignores bits 15:14) / CHV, BSW
3 **Programming Notes:** See the programming note in the previous DWORD for the Min values. The clear rectangle x and y min and max values must be shifted by the LOD level; i.e. the hardware does not include the LOD in this function. Hence to clear any particular X, Y from the base level, to clear the contents at level "LOD" use (X»LOD) and (Y»LOD). Hence the max values must be less than or equal to: (Surface Width » LOD) and (Surface Height » LOD) for X Max and Y Max respectively.	31:16	**Clear Rectangle Y Max** Project: All Format: U16 in Pixels from Depth Buffer origin (lower right corner) Specifies Ymax value of (exclusive) of clear rectangle with the Depth Buffer, used for clipping. Pixels with Y coordinates greater than Ymax will be not be cleared. **Value** / **Name** / **Project** [0,16383] / (Device ignores bits 31:30) / CHV, BSW
	15:0	**Clear Rectangle X Max** Project: All Format: U16 in Pixels from Depth Buffer origin (lower right corner) Specifies Xmax value of (exclusive) of clear rectangle with the Depth Buffer, used for clipping. Pixels with X coordinates greater than or equal to Xmax will be not be affected. **Value** / **Name** / **Project** [0,16383] / (Device ignores bits 15:14) / CHV, BSW
4	31:16	**Reserved** Project: All Format: MBZ
	15:0	**Sample Mask**

3DSTATE_WM_HZ_OP		
		Project: All
		Format: Right-justified bitmask (Bit 0 = Sample0). Number of bits that are used is determined by Num Multisamples (3DSTATE_WM_HZ_OP)
		A per-multisample-position mask state variable that is immediately and unconditionally ANDed with the sample coverage mask as part of the rasterization process. This mask is applied prior to centroid selection.
		Programming Notes
		If Number of Multisamples is NUMSAMPLES_1, bits 15:1 of this field will be zeroed by HW.If Number of Multisamples is NUMSAMPLES_2, bits 15:2 of this field will be zeroed by HW.If Number of Multisamples is NUMSAMPLES_4, bits 15:4 of this field will be zeroed by HW.If Number of Multisamples is NUMSAMPLES_8, bits 15:8 of this field will be zeroed by HW.

A64 Byte Scattered Write MSD

MSD1W_A64_BS - A64 Byte Scattered Write MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Scattered R/W
Group:	Byte Scattered R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 1Ah Project: All Format: Opcode A64 Scattered Write message
	13	**Reserved**
	12	**Reserved**
	11:10	**Data Elements** Project: All Format: MDC_A64_DS [CHV, BSW] Specifies the number of data elements to be read or written
	9:8	**A64 Scattered Message Subtype** Default Value: 0h Project: All Format: Opcode Byte Read/Write subtype
	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Dword Scattered Read MSD

MSD1R_A64_DWS - A64 Dword Scattered Read MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Scattered R/W
Group:	DW Scattered R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 10h Project: All Format: Opcode A64 Scattered Read message
	13	**Invalidate After Read** Project: All Format: MDC_IAR [CHV, BSW] Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12	**Reserved**
	11:10	**Data Elements** Project: All Format: MDC_A64_DS [CHV, BSW] Specifies the number of data elements to be read or written
	9:8	**A64 Scattered Message Subtype** Default Value: 1h Project: All Format: Opcode Dword Read/Write subtype
	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Dword Scattered Write MSD

MSD1W_A64_DWS - A64 Dword Scattered Write MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Scattered R/W
Group:	DW Scattered R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 1Ah Project: All Format: Opcode A64 Scattered Write message
	13	**Reserved** Project: All Format: MBZ Ignored
	12	**Reserved** Project: CHV, BSW Format: Ignore Ignored
	11:10	**Data Elements** Project: All Format: MDC_A64_DS [CHV, BSW] Specifies the number of data elements to be read or written
	9:8	**A64 Scattered Message Subtype** Default Value: 1h Project: All Format: Opcode Dword Read/Write subtype

MSD1W_A64_DWS - A64 Dword Scattered Write MSD

	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Dword SIMD4x2 Untyped Atomic Integer Binary with Return Data Operation MSD

MSD1R_A64_DWAI2_4x2 - A64 Dword SIMD4x2 Untyped Atomic Integer Binary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Untyped Atomic Integer Binary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] The message forbids a header
	18:14	**Message Type** Default Value: 13h Project: All Format: Opcode A64 Untyped Atomic Integer Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**Data Width** Default Value: 0h Project: All Format: Opcode Operations are on 32-bit integers
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP2 [CHV, BSW] Specifies the atomic integer operation to be performed.

MSD1R_A64_DWAI2_4x2 - A64 Dword SIMD4x2 Untyped Atomic Integer Binary with Return Data Operation MSD

	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Dword SIMD4x2 Untyped Atomic Integer Binary Write Only Operation MSD

MSD1W_A64_DWAI2_4x2 - A64 Dword SIMD4x2 Untyped Atomic Integer Binary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Untyped Atomic Integer Binary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] The message forbids a header
	18:14	**Message Type** Default Value: 13h Project: All Format: Opcode A64 Untyped Atomic Integer Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Data Width** Default Value: 0h Project: All Format: Opcode Operations are on 32-bit integers
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP2 [CHV, BSW] Specifies the atomic integer operation to be performed.

MSD1W_A64_DWAI2_4x2 - A64 Dword SIMD4x2 Untyped Atomic Integer Binary Write Only Operation MSD		
	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Dword SIMD4x2 Untyped Atomic Integer Trinary with Return Data Operation MSD

MSD1R_A64_DWAI3_4x2 - A64 Dword SIMD4x2 Untyped Atomic Integer Trinary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Untyped Atomic Integer Trinary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] The message forbids a header
	18:14	**Message Type** Default Value: 13h Project: All Format: Opcode A64 Untyped Atomic Integer Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**Data Width** Default Value: 0h Project: All Format: Opcode Operations are on 32-bit integers
	11:8	**Atomic Integer Operation** Project: CHV, BSW Format: MDC_AOP3S [CHV, BSW] Specifies the atomic integer operation to be performed.

MSD1R_A64_DWAI3_4x2 - A64 Dword SIMD4x2 Untyped Atomic Integer Trinary with Return Data Operation MSD

		Workaround CMPWR_2W Operation is not supported in A64 SIMD4x2.
	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Dword SIMD4x2 Untyped Atomic Integer Trinary Write Only Operation MSD

MSD1W_A64_DWAI3_4x2 - A64 Dword SIMD4x2 Untyped Atomic Integer Trinary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Untyped Atomic Integer Trinary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] The message forbids a header
	18:14	**Message Type** Default Value: 13h Project: All Format: Opcode A64 Untyped Atomic Integer Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Data Width** Default Value: 0h Project: All Format: Opcode Operations are on 32-bit integers
	11:8	**Atomic Integer Operation** Project: CHV, BSW Format: MDC_AOP3S [CHV, BSW] Specifies the atomic integer operation to be performed.

MSD1W_A64_DWAI3_4x2 - A64 Dword SIMD4x2 Untyped Atomic Integer Trinary Write Only Operation MSD

		Workaround CMPWR_2W is not supported by A64 SIMD4x2.
	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Dword SIMD4x2 Untyped Atomic Integer Unary with Return Data Operation MSD

MSD1R_A64_DWAI1_4x2 - A64 Dword SIMD4x2 Untyped Atomic Integer Unary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Untyped Atomic Integer Unary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] The message forbids a header
	18:14	**Message Type** Default Value: 13h Project: All Format: Opcode A64 Untyped Atomic Integer Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**Data Width** Default Value: 0h Project: All Format: Opcode Operations are on 32-bit integers
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP1 [CHV, BSW] Specifies the atomic integer operation to be performed.

MSD1R_A64_DWAI1_4x2 - A64 Dword SIMD4x2 Untyped Atomic Integer Unary with Return Data Operation MSD		
	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Dword SIMD4x2 Untyped Atomic Integer Unary Write Only Operation MSD

MSD1W_A64_DWAI1_4x2 - A64 Dword SIMD4x2 Untyped Atomic Integer Unary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Untyped Atomic Integer Unary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] The message forbids a header
	18:14	**Message Type** Default Value: 13h Project: All Format: Opcode A64 Untyped Atomic Integer Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Data Width** Default Value: 0h Project: All Format: Opcode Operations are on 32-bit integers
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP1 [CHV, BSW] Specifies the atomic integer operation to be performed.

MSD1W_A64_DWAI1_4x2 - A64 Dword SIMD4x2 Untyped Atomic Integer Unary Write Only Operation MSD		
	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Dword Untyped Atomic Integer Binary with Return Data Operation MSD

MSD1R_A64_DWAI2 - A64 Dword Untyped Atomic Integer Binary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Untyped Atomic Integer Binary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] The message forbids a header
	18:14	**Message Type** Default Value: 12h Project: All Format: Opcode A64 Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**Data Width** Default Value: 0h Project: All Format: Opcode Operations are on 32-bit integers
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP2 [CHV, BSW] Specifies the atomic integer operation to be performed.

MSD1R_A64_DWAI2 - A64 Dword Untyped Atomic Integer Binary with Return Data Operation MSD		
	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Dword Untyped Atomic Integer Binary Write Only Operation MSD

MSD1W_A64_DWAI2 - A64 Dword Untyped Atomic Integer Binary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Untyped Atomic Integer Binary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] The message forbids a header
	18:14	**Message Type** Default Value: 12h Project: All Format: Opcode A64 Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Data Width** Default Value: 0h Project: All Format: Opcode Operations are on 32-bit integers
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP2 [CHV, BSW] Specifies the atomic integer operation to be performed.

MSD1W_A64_DWAI2 - A64 Dword Untyped Atomic Integer Binary Write Only Operation MSD

	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Dword Untyped Atomic Integer Trinary with Return Data Operation MSD

MSD1R_A64_DWAI3 - A64 Dword Untyped Atomic Integer Trinary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Untyped Atomic Integer Trinary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] The message forbids a header
	18:14	**Message Type** Default Value: 12h Project: All Format: Opcode A64 Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**Data Width** Default Value: 0h Project: All Format: Opcode Operations are on 32-bit integers
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP3 [CHV, BSW] Specifies the atomic integer operation to be performed.

MSD1R_A64_DWAI3 - A64 Dword Untyped Atomic Integer Trinary with Return Data Operation MSD

	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Dword Untyped Atomic Integer Trinary Write Only Operation MSD

MSD1W_A64_DWAI3 - A64 Dword Untyped Atomic Integer Trinary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Untyped Atomic Integer Trinary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] The message forbids a header
	18:14	**Message Type** Default Value: 12h Project: All Format: Opcode A64 Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Data Width** Default Value: 0h Project: All Format: Opcode Operations are on 32-bit integers
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP3 [CHV, BSW] Specifies the atomic integer operation to be performed.

MSD1W_A64_DWAI3 - A64 Dword Untyped Atomic Integer Trinary Write Only Operation MSD

	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Dword Untyped Atomic Integer Unary with Return Data Operation MSD

MSD1R_A64_DWAI1 - A64 Dword Untyped Atomic Integer Unary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Untyped Atomic Integer Unary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] The message forbids a header
	18:14	**Message Type** Default Value: 12h Project: All Format: Opcode A64 Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**Data Width** Default Value: 0h Project: All Format: Opcode Operations are on 32-bit integers
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP1 [CHV, BSW] Specifies the atomic integer operation to be performed.

MSD1R_A64_DWAI1 - A64 Dword Untyped Atomic Integer Unary with Return Data Operation MSD

	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Dword Untyped Atomic Integer Unary Write Only Operation MSD

MSD1W_A64_DWAI1 - A64 Dword Untyped Atomic Integer Unary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Untyped Atomic Integer Unary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] The message forbids a header
	18:14	**Message Type** Default Value: 12h Project: All Format: Opcode A64 Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Data Width** Default Value: 0h Project: All Format: Opcode Operations are on 32-bit integers
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP1 [CHV, BSW] Specifies the atomic integer operation to be performed.

MSD1W_A64_DWAI1 - A64 Dword Untyped Atomic Integer Unary Write Only Operation MSD		
	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Hword Block Read MSD

MSD1R_A64_HWB - A64 Hword Block Read MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Block R/W
Group:	HW Block R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 14h Project: All Format: Opcode A64 Oword Block Read message
	13	**Invalidate After Read** Project: All Format: MDC_IAR [CHV, BSW] Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12:11	**A64 Block Message Subtype** Default Value: 3h Project: All Format: Opcode Hword Block Read/Write subtype
	10:8	**Data Elements** Project: All Format: MDC_A64_DB_HW [CHV, BSW] Specifies the number of contiguous Hwords to be read or written
	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Hword Block Write MSD

MSD1W_A64_HWB - A64 Hword Block Write MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Block R/W
Group:	HW Block R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 15h Project: All Format: Opcode A64 Hword Block Write message
	13	**Reserved** Project: All Format: MBZ Ignored
	12:11	**A64 Block Message Subtype** Default Value: 3h Project: All Format: Opcode Hword Block Read/Write subtype
	10:8	**Data Elements** Project: All Format: MDC_A64_DB_HW [CHV, BSW] Specifies the number of contiguous Hwords to be read or written
	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Oword Block Read MSD

MSD1R_A64_OWB - A64 Oword Block Read MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Block R/W
Group:	OW Block R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 14h Project: All Format: Opcode A64 Oword Block Read message
	13	**Invalidate After Read** Project: All Format: MDC_IAR [CHV, BSW] Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12:11	**A64 Block Message Subtype** Default Value: 0h Project: All Format: Opcode Oword Block Read/Write subtype
	10:8	**Data Elements** Project: All Format: MDC_A64_DB_OW [CHV, BSW] Specifies the number of contiguous Owords to be read or written
	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Oword Block Write MSD

MSD1W_A64_OWB - A64 Oword Block Write MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Block R/W
Group:	OW Block R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 15h Project: All Format: Opcode A64 Oword Block Write message
	13	**Reserved** Project: All Format: MBZ Ignored
	12:11	**A64 Block Message Subtype** Default Value: 0h Project: All Format: Opcode Oword Block Read/Write subtype
	10:8	**Data Elements** Project: All Format: MDC_A64_DB_OW [CHV, BSW] Specifies the number of contiguous Owords to be read or written
	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Oword Dual Block Read MSD

MSD1R_A64_OWDB - A64 Oword Dual Block Read MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Block R/W
Group:	OW Dual Block R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 14h Project: All Format: Opcode A64 Oword Block Read message
	13	**Invalidate After Read** Project: All Format: MDC_IAR [CHV, BSW] Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12:11	**A64 Block Message Subtype** Default Value: 2h Project: All Format: Opcode Oword Dual Block Read/Write subtype
	10:8	**Data Elements** Project: All Format: MDC_A64_DB_OWD Specifies the number of contiguous Owords to be read or written
	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Oword Dual Block Write MSD

MSD1W_A64_OWDB - A64 Oword Dual Block Write MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Block R/W
Group:	OW Dual Block R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 15h Project: All Format: Opcode A64 Oword Block Write message
	13	**Reserved** Project: All Format: MBZ Ignored
	12:11	**A64 Block Message Subtype** Default Value: 2h Project: All Format: Opcode Oword Dual Block Read/Write subtype
	10:8	**Data Elements** Project: All Format: MDC_A64_DB_OWD Specifies the number of contiguous Owords to be read or written
	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Oword Unaligned Block Read MSD

MSD1R_A64_OWUB - A64 Oword Unaligned Block Read MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Block R/W
Group:	OW Unaligned Block R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 14h Project: All Format: Opcode A64 Oword Block Read message
	13	**Reserved** Project: All Format: MBZ Ignored
	12:11	**A64 Block Message Subtype** Default Value: 1h Project: All Format: Opcode Oword Unaligned Block Read subtype
	10:8	**Data Elements** Project: All Format: MDC_A64_DB_OW [CHV, BSW] Specifies the number of contiguous Owords to be read
	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Qword Scattered Write MSD

MSD1W_A64_QWS - A64 Qword Scattered Write MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Scattered R/W
Group:	QW Scattered R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 1Ah Project: All Format: Opcode A64 Scattered Write message
	13	**Reserved** Project: All Format: MBZ Ignored
	12	**Reserved** Project: CHV, BSW Format: Ignore Ignored
	11:10	**Data Elements** Project: All Format: MDC_A64_DS [CHV, BSW] Specifies the number of data elements to be read or written
	9:8	**A64 Scattered Message Subtype** Default Value: 2h Project: All Format: Opcode Qword Read/Write subtype

MSD1W_A64_QWS - A64 Qword Scattered Write MSD		
	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Qword SIMD4x2 Untyped Atomic Integer Binary with Return Data Operation MSD

MSD1R_A64_QWAI2_4x2 - A64 Qword SIMD4x2 Untyped Atomic Integer Binary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Qword Untyped Atomic Integer Binary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] The message forbids a header
	18:14	**Message Type** Default Value: 13h Project: All Format: Opcode A64 Untyped Atomic Integer Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**Data Width** Default Value: 1h Project: All Format: Opcode Operations are on 64-bit integers
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP2 [CHV, BSW] Specifies the atomic integer operation to be performed.

MSD1R_A64_QWAI2_4x2 - A64 Qword SIMD4x2 Untyped Atomic Integer Binary with Return Data Operation MSD

<table>
<tr><td></td><td>7:0</td><td>Binding Table Index

Project:	All
Format:	MDC_STATELESS [CHV, BSW]

Specifies the message is stateless</td></tr>
</table>

A64 Qword SIMD4x2 Untyped Atomic Integer Binary Write Only Operation MSD

MSD1W_A64_QWAI2_4x2 - A64 Qword SIMD4x2 Untyped Atomic Integer Binary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Qword Untyped Atomic Integer Binary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] The message forbids a header
	18:14	**Message Type** Default Value: 13h Project: All Format: Opcode A64 Untyped Atomic Integer Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Data Width** Default Value: 1h Project: All Format: Opcode Operations are on 64-bit integers
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP2 [CHV, BSW] Specifies the atomic integer operation to be performed.

MSD1W_A64_QWAI2_4x2 - A64 Qword SIMD4x2 Untyped Atomic Integer Binary Write Only Operation MSD

	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Qword SIMD4x2 Untyped Atomic Integer Trinary with Return Data Operation MSD

MSD1R_A64_QWAI3_4x2 - A64 Qword SIMD4x2 Untyped Atomic Integer Trinary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Qword Untyped Atomic Integer Trinary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] The message forbids a header
	18:14	**Message Type** Default Value: 13h Project: All Format: Opcode A64 Untyped Atomic Integer Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**Data Width** Default Value: 1h Project: All Format: Opcode Operations are on 64-bit integers
	11:8	**Atomic Integer Operation** Project: CHV, BSW Specifies the atomic integer operation to be performed. **Workaround** CMPWR_2W is not supported in A64 SIMD4x2

MSD1R_A64_QWAI3_4x2 - A64 Qword SIMD4x2 Untyped Atomic Integer Trinary with Return Data Operation MSD

	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Qword SIMD4x2 Untyped Atomic Integer Trinary Write Only Operation MSD

MSD1W_A64_QWAI3_4x2 - A64 Qword SIMD4x2 Untyped Atomic Integer Trinary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Qword Untyped Atomic Integer Trinary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] The message forbids a header
	18:14	**Message Type** Default Value: 13h Project: All Format: Opcode A64 Untyped Atomic Integer Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Data Width** Default Value: 1h Project: All Format: Opcode Operations are on 64-bit integers
	11:8	**Atomic Integer Operation** Project: CHV, BSW Format: MDC_AOP3S [CHV, BSW] Specifies the atomic integer operation to be performed.

MSD1W_A64_QWAI3_4x2 - A64 Qword SIMD4x2 Untyped Atomic Integer Trinary Write Only Operation MSD

		Workaround CMPWR_2W is not supported in A64 SIMD4x2.
	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Qword SIMD4x2 Untyped Atomic Integer Unary with Return Data Operation MSD

MSD1R_A64_QWAI1_4x2 - A64 Qword SIMD4x2 Untyped Atomic Integer Unary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Qword Untyped Atomic Integer Unary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] The message forbids a header
	18:14	**Message Type** Default Value: 13h Project: All Format: Opcode A64 Untyped Atomic Integer Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**Data Width** Default Value: 1h Project: All Format: Opcode Operations are on 64-bit integers
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP1 [CHV, BSW] Specifies the atomic integer operation to be performed.

MSD1R_A64_QWAI1_4x2 - A64 Qword SIMD4x2 Untyped Atomic Integer Unary with Return Data Operation MSD

	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Qword SIMD4x2 Untyped Atomic Integer Unary Write Only Operation MSD

MSD1W_A64_QWAI1_4x2 - A64 Qword SIMD4x2 Untyped Atomic Integer Unary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Qword Untyped Atomic Integer Unary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] The message forbids a header
	18:14	**Message Type** Default Value: 13h Project: All Format: Opcode A64 Untyped Atomic Integer Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Data Width** Default Value: 1h Project: All Format: Opcode Operations are on 64-bit integers
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP1 [CHV, BSW] Specifies the atomic integer operation to be performed.

MSD1W_A64_QWAI1_4x2 - A64 Qword SIMD4x2 Untyped Atomic Integer Unary Write Only Operation MSD

	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Qword Untyped Atomic Integer Binary with Return Data Operation MSD

MSD1R_A64_QWAI2 - A64 Qword Untyped Atomic Integer Binary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Qword Untyped Atomic Integer Binary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] The message forbids a header
	18:14	**Message Type** Default Value: 12h Project: All Format: Opcode A64 Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**Data Width** Default Value: 1h Project: All Format: Opcode Operations are on 64-bit integers
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP2 [CHV, BSW] Specifies the atomic integer operation to be performed.

MSD1R_A64_QWAI2 - A64 Qword Untyped Atomic Integer Binary with Return Data Operation MSD		
	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Qword Untyped Atomic Integer Binary Write Only Operation MSD

MSD1W_A64_QWAI2 - A64 Qword Untyped Atomic Integer Binary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Qword Untyped Atomic Integer Binary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] The message forbids a header
	18:14	**Message Type** Default Value: 12h Project: All Format: Opcode A64 Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Data Width** Default Value: 1h Project: All Format: Opcode Operations are on 64-bit integers
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP2 [CHV, BSW] Specifies the atomic integer operation to be performed.

MSD1W_A64_QWAI2 - A64 Qword Untyped Atomic Integer Binary Write Only Operation MSD

	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Qword Untyped Atomic Integer Trinary with Return Data Operation MSD

MSD1R_A64_QWAI3 - A64 Qword Untyped Atomic Integer Trinary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Qword Untyped Atomic Integer Trinary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] The message forbids a header
	18:14	**Message Type** Default Value: 12h Project: All Format: Opcode A64 Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**Data Width** Default Value: 1h Project: All Format: Opcode Operations are on 64-bit integers
	11:8	**Atomic Integer Operation** Project: CHV, BSW Format: MDC_AOP3S [CHV, BSW] Specifies the atomic integer operation to be performed.

MSD1R_A64_QWAI3 - A64 Qword Untyped Atomic Integer Trinary with Return Data Operation MSD

<table>
<tr><td></td><td></td><td><table><tr><th>Workaround</th></tr><tr><td>CMPWR_2W is not supported in A64 QWord SIMD8.</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td>Binding Table Index<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MDC_STATELESS [CHV, BSW]</td></tr></table>Specifies the message is stateless</td></tr>
</table>

A64 Qword Untyped Atomic Integer Trinary Write Only Operation MSD

MSD1W_A64_QWAI3 - A64 Qword Untyped Atomic Integer Trinary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Qword Untyped Atomic Integer Trinary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] The message forbids a header
	18:14	**Message Type** Default Value: 12h Project: All Format: Opcode A64 Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Data Width** Default Value: 1h Project: All Format: Opcode Operations are on 64-bit integers
	11:8	**Atomic Integer Operation** Project: CHV, BSW Format: MDC_AOP3S [CHV, BSW] Specifies the atomic integer operation to be performed.

MSD1W_A64_QWAI3 - A64 Qword Untyped Atomic Integer Trinary Write Only Operation MSD

<table>
<tr><td></td><td></td><td><table><tr><th>Workaround</th></tr><tr><td>CMPWR_2W is not supported in A64 QWord SIMD8.</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td>Binding Table Index<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MDC_STATELESS [CHV, BSW]</td></tr></table>Specifies the message is stateless</td></tr>
</table>

A64 Qword Untyped Atomic Integer Unary with Return Data Operation MSD

MSD1R_A64_QWAI1 - A64 Qword Untyped Atomic Integer Unary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Qword Untyped Atomic Integer Unary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] The message forbids a header
	18:14	**Message Type** Default Value: 12h Project: All Format: Opcode A64 Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**Data Width** Default Value: 1h Project: All Format: Opcode Operations are on 64-bit integers
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP1 [CHV, BSW] Specifies the atomic integer operation to be performed.

MSD1R_A64_QWAI1 - A64 Qword Untyped Atomic Integer Unary with Return Data Operation MSD

	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Qword Untyped Atomic Integer Unary Write Only Operation MSD

MSD1W_A64_QWAI1 - A64 Qword Untyped Atomic Integer Unary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Qword Untyped Atomic Integer Unary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] The message forbids a header
	18:14	**Message Type** Default Value: 12h Project: All Format: Opcode A64 Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Data Width** Default Value: 1h Project: All Format: Opcode Operations are on 64-bit integers
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP1 [CHV, BSW] Specifies the atomic integer operation to be performed.

MSD1W_A64_QWAI1 - A64 Qword Untyped Atomic Integer Unary Write Only Operation MSD

	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Untyped Surface Read MSD

MSD1R_A64_US - A64 Untyped Surface Read MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Surface R/W
Group:	Scattered Untyped Surface R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 11h Project: All Format: Opcode A64 Untyped Surface Read message
	13:12	**SIMD Mode** Project: All Format: MDC_SM3 [CHV, BSW] Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Channel Mask** Project: All Format: MDC_CMASK [CHV, BSW] Specifies which RGBA channels are included in the message payload.
	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

A64 Untyped Surface Write MSD

MSD1W_A64_US - A64 Untyped Surface Write MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Surface R/W
Group:	Scattered Untyped Surface R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHF [CHV, BSW] Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 19h Project: All Format: Opcode A64 Untyped Surface Write message
	13:12	**SIMD Mode** Project: All Format: MDC_SM3 [CHV, BSW] Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Channel Mask** Project: All Format: MDC_UW_CMASK [CHV, BSW] Specifies which RGBA channels are included in the message payload.
	7:0	**Binding Table Index** Project: All Format: MDC_STATELESS [CHV, BSW] Specifies the message is stateless

Addition

add - Addition

Project:	CHV, BSW
Source:	Eulsa
Length Bias:	4

The add instruction performs component-wise addition of src0 and src1 and stores the results in dst. Addition of two floating-point numbers follows rules in add (IEEE mode) or add (ALT mode).

Format: [(pred)] add[.cmod] (exec_size) dst src0 src1

Programming Notes
Use a source modifier with add to implement subtraction.

Syntax
[(pred)] add[.cmod] (exec_size) reg reg reg [(pred)] add[.cmod] (exec_size) reg reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = src0.chan[n] + src1.chan[n]; } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types	Project
*B,*W,*D	*B,*W,*D	
*B,*W,*D	F	
F	F	
DF	DF	CHV, BSW
HF	HF	CHV, BSW
*B,*W,*D	HF	CHV, BSW
*W,*D,*Q	*W,*D,*Q	CHV, BSW
HF, F	HF, F	CHV, BSW

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM [CHV, BSW]

add - Addition		
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Addition with Carry

addc - Addition with Carry

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The addc instruction performs component-wise addition of src0 and src1 and stores the results in dst; it also stores the carry into acc. If the operation produces a carry out, 0x00000001 is stored in acc, else 0x00000000 is stored in acc.

Format: [(pred)] addc[.cmod] (exec_size) dst src0 src1

Restriction
AccWrEn is required. The accumulator is an implicit destination and thus cannot be an explicit destination operand.

Syntax
[(pred)] addc[.cmod] (exec_size) reg reg reg [(pred)] addc[.cmod] (exec_size) reg reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = src0.chan[n] + src1.chan[n]; acc.chan[n] = carry(src0.chan[n] + src1.chan[n]); } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	N	N

Src Types	Dst Types
UD	UD

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Arithmetic Shift Right

asr - Arithmetic Shift Right

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

Perform component-wise arithmetic right shift of the bits in src0 by the shift count indicated in src1, storing the results in dst. If src0 has a signed type, insert copies of src0's sign bit in the number of MSBs indicated by the shift count. Otherwise insert 0 bits. In QWord mode, the shift count is taken from the low six bits of src1 regardless of the src1 type and treated as an unsigned integer in the range 0 to 63. Otherwise the shift count is taken from the low five bits of src1 regardless of the src1 type and treated as an unsigned integer in the range 0 to 31. The operation uses QWord mode if src0 or dst has the Q or UQ type but not if src1 is the only operand with the Q or UQ type. For positive values, this operation is src0 / 2shiftCount and for negative values, this operation is src0 / 2shiftCount - 1.

Format: [(pred)] asr[.cmod] (exec_size) dst src0 src1

Programming Notes
If src0 is -1, the result is -1 regardless of the shift count.
For unsigned src0 types, asr and shr produce the same result.

Syntax
[(pred)] asr[.cmod] (exec_size) reg reg reg [(pred)] asr[.cmod] (exec_size) reg reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.channel[n]) { shiftCnt = src0 or dst has Q or UQ type ? src1.chan[n] & 0x3F : src1.chan[n] & 0x1F if (src0.chan[n] >= 0) { dst.chan[n] = src0.chan[n] » shiftCnt; } else { int maskLSB = pow(2, shiftCnt) - 1; if (maskLSB & src0.chan[n] == 0) { dst.chan[n] = sign(src0.chan[n]) * ((abs)src0.chan[n] » shiftCnt); } else { dst.chan[n] = sign(src0.chan[n]) * ((abs)src0.chan[n] » shiftCnt) - 1; } } } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types	Project
*B,*W,*D	*B,*W,*D	
*W,*D,*Q	*W,*D,*Q	CHV, BSW

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM [CHV, BSW]

asr - Arithmetic Shift Right		
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Average

avg - Average

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The avg instruction performs component-wise integer average of src0 and src1 and stores the results in dst. An integer average uses integer upward rounding. It is equivalent to increment one to the addition of src0 and src1 and then apply an arithmetic right shift to this intermediate value.

Format: The avg instruction performs component-wise integer average of src0 and src1 and stores the results in dst. An integer average uses integer upward rounding. It is equivalent to increment one to the addition of src0 and src1 and then apply an arithmetic right shift to this intermediate value.

Syntax
[(pred)] avg[.cmod] (exec_size) reg reg reg [(pred)] avg[.cmod] (exec_size) reg reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = (src0.chan[n] + src1.chan[n] + 1) » 1; // Use arithmetic shift right. } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types
*B,*W,*D	*B,*W,*D

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Bit Field Extract

bfe - Bit Field Extract

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

Component-wise extract a bit field from src2 using the bit field width from src0 and the bit field offset from src1. Store the extracted bit field value in the low bits of dst and sign extend (if D type) or zero extend (if UD type). The width and offset values are from the low five bits of src0 and src1 respectively, or src0 & 0x1f and src1 & 0x1f. If width is zero, the result is zero. If offset + width > 32 then the extracted bit field is bits offset to 31 of src2, extracting only 32 - offset bits, less than width as the bit field cannot extend past the MSB of the source value. Otherwise extract width bits extending from bit positions offset to offset + width - 1.

Format: [(pred)] bfe (exec_size) dst src0 src1 src2

Restriction
No accumulator access, implicit or explicit.
All three-source instructions have certain restrictions, described in Instruction Formats [CHV, BSW].

Syntax
[(pred)] bfe (exec_size) reg reg reg reg

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { UD width = src0.chan[n][4:0]; UD offset = src1.chan[n][4:0]; if (width == 0) { dst.chan[n] = 0x00000000; } else if ((width + offset) < 32) { dst.chan[n] = src2.chan[n] « (32 - width - offset); if (src2 is signed) { dst.chan[n] = dst.chan[n] » (32 - width); // pad sign bit of dst.chan } else { dst.chan[n] = dst.chan[n] » (32 - width); // pad 0 } } else { if (src2 is signed) { dst.chan[n] = src2.chan[n] » offset; // pad sign bit } else { dst.chan[n] = src2.chan[n] » offset; // pad 0 } } } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

Src Types	Dst Types
UD	UD
D	D

DWord	Bit	Description
0..3	127:126	**Reserved** Format: MBZ
	125:106	**Source 2** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC [CHV, BSW]
	105	**Reserved** Format: MBZ

bfe - Bit Field Extract

	104:85	**Source 1** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC [CHV, BSW]
	84	**Reserved** Format: MBZ
	83:64	**Source 0** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC [CHV, BSW]
	63:56	**Destination Register Number** Format: DstRegNum [CHV, BSW]
	55:53	**Destination Subregister Number** Format: DstSubRegNum[2:0]
	52:49	**Destination Channel Enable** Format: ChanEn[4] Four channel enables are defined for controlling which channels are written into the destination region. These channel mask bits are applied in a modulo-four manner to all ExecSize channels. There is 1-bit Channel Enable for each channel within the group of 4. If the bit is cleared, the write for the corresponding channel is disabled. If the bit is set, the write is enabled. Mnemonics for the bit being set for the group of 4 are x, y, z, and w, respectively, where x corresponds to Channel 0 in the group and w corresponds to channel 3 in the group
	48:42	**Reserved** Project: CHV, BSW Format: MBZ
	41:40	**Source 2 Modifier** Exists If: //([Property[Source Modifier]=='true') Format: SrcMod [CHV, BSW]
	39:38	**Source 1 Modifier** Exists If: //([Property[Source Modifier]=='true') Format: SrcMod [CHV, BSW]
	41:36	**Reserved** Exists If: //([Property[Source Modifier]=='false') Format: MBZ
	37:36	**Source 0 Modifier** Exists If: //([Property[Source Modifier]=='true') Format: SrcMod [CHV, BSW]
	35	**Reserved** Format: MBZ
	34	**Reserved**

bfe - Bit Field Extract		
	33	**Flag Subregister Number** This field contains the flag subregister number for instructions with a non-zero Conditional Modifier.
	32	**Reserved** Format: MBZ
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Bit Field Insert 1

bfi1 - Bit Field Insert 1

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The bfi1 instruction is the first instruction in a two-instruction macro for bfi (Bit Field Insert). The bfi1 instruction component-wise generates mask with control from src0 and src1 and stores the results in dst. The mask is used in the bfi2 instruction to generate the final result of bfi. Create a bit mask corresponding to the bit field width and offset in src0 and src1. Store the bit mask in dst. The mask has all bits in the bit field set to 1 and all other bits as 0. The width and offset values are from the low five bits of src0 and src1 respectively, or src0 & 0x1f and src1 & 0x1f. If width is zero, the result is zero. The bfi macro has four source operands: src0 - bit field width in low five bits, src1 - bit field offset/starting bit position in low five bits, src2 - bit field value to insert, using only the number of least significant bits given by width in src0, and src3 - overall value into which the bit field is inserted, providing all bits other than the inserted bits for the result value. bfi dst src0 src1 src2 src3 // Translates to these two instructions: bfi1 dst src0 src1 bfi2 dst dst src2 src3

Format: [(pred)] bfi1 (exec_size) dst src0 src1

Programming Notes
No accumulator access, implicit or explicit.

Syntax
[(pred)] bfi1 (exec_size) reg reg reg [(pred)] bfi1 (exec_size) reg reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { UD width = src0.chan[n][4:0]; UD offset = src1.chan[n][4:0]; dst = ((1 « width) - 1) « offset; } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

Src Types	Dst Types
UD	UD
D	D

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM [CHV, BSW]

bfi1 - Bit Field Insert 1		
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Bit Field Insert 2

bfi2 - Bit Field Insert 2

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The bfi2 instruction is the second instruction in a two-instruction macro for bfi (Bit Field Insert). The bfi2 instruction component-wise performs the bitfield insert operation on src1 and src2 based on the mask in src0. Use the mask in src0 to take a bit field value from the low bits of src1 and combine it with the value from src2 (so src2 provides all bits other than those masked out and replaced by the bit field value). Store the result in dst. The bfi macro has four source operands: src0 - bit field width in low five bits, src1 - bit field offset/starting bit position in low five bits, src2 - bit field value to insert, using only the number of least significant bits given by width in src0, and src3 - overall value into which the bit field is inserted, providing all bits other than the inserted bits for the result value. bfi dst src0 src1 src2 src3 // Translates to these two instructions: bfi1 dst src0 src1 bfi2 dst dst src2 src3

Format: [(pred)] bfi2 (exec_size) dst src0 src1 src2

Restriction	Project
No accumulator access, implicit or explicit.	
All three-source instructions have certain restrictions, described in Instruction Formats [CHV, BSW].	CHV, BSW

Syntax
[(pred)] bfi2 (exec_size) reg reg reg reg

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { UD offset = LZD(reverse(src0.chan[n]))-1; // offset is the number of LSB zero bits below the bit mask which has all 1s. // width (implied by the logic) is the number of 1 bits in the mask value, which should be all 1s. dst.chan[n] = ((src1.chan[n] « offset) & src0.chan[n]) | (src2.chan[n] & ! src0.chan[n]); }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

Src Types	Dst Types
UD	UD
D	D

DWord	Bit	Description
0..3	127:126	**Reserved** Format: MBZ
	125:106	**Source 2** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC [CHV, BSW]

bfi2 - Bit Field Insert 2

	105	**Reserved** Format: MBZ
	104:85	**Source 1** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC [CHV, BSW]
	84	**Reserved** Format: MBZ
	83:64	**Source 0** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC [CHV, BSW]
	63:56	**Destination Register Number** Format: DstRegNum [CHV, BSW]
	55:53	**Destination Subregister Number** Format: DstSubRegNum[2:0]
	52:49	**Destination Channel Enable** Format: ChanEn[4] Four channel enables are defined for controlling which channels are written into the destination region. These channel mask bits are applied in a modulo-four manner to all ExecSize channels. There is 1-bit Channel Enable for each channel within the group of 4. If the bit is cleared, the write for the corresponding channel is disabled. If the bit is set, the write is enabled. Mnemonics for the bit being set for the group of 4 are x, y, z, and w, respectively, where x corresponds to Channel 0 in the group and w corresponds to channel 3 in the group
	48:42	**Reserved** Format: MBZ
	41:40	**Source 2 Modifier** Exists If: //([Property[Source Modifier]=='true') Format: SrcMod [CHV, BSW]
	39:38	**Source 1 Modifier** Exists If: //([Property[Source Modifier]=='true') Format: SrcMod [CHV, BSW]
	41:36	**Reserved** Exists If: //([Property[Source Modifier]=='false') Format: MBZ
	37:36	**Source 0 Modifier** Exists If: //([Property[Source Modifier]=='true') Format: SrcMod [CHV, BSW]
	35	**Reserved** Format: MBZ

bfi2 - Bit Field Insert 2

	34	**Reserved** Format: MBZ
	33	**Flag Subregister Number** This field contains the flag subregister number for instructions with a non-zero Conditional Modifier.
	32	**Reserved** Format: MBZ
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Bit Field Reverse

bfrev - Bit Field Reverse

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The bfrev instruction component-wise reverses all the bits in src0 and stores the results in dst.

Format: [(pred)] bfrev (exec_size) dst src0

Restriction
No accumulator access, implicit or explicit.

Syntax
[(pred)] bfrev (exec_size) reg reg [(pred)] bfrev (exec_size) reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { for (idx = 0; idx < 32; idx++) { dst.chan[n][idx] = src0.chan[n][31-idx]; } } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

Src Types	Dst Types
UD	UD

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32 [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Branch Converging

brc - Branch Converging

Project:	CHV, BSW
Source:	Eulsa
Length Bias:	4

Description
The brc instruction redirects the execution forward or backward to the instruction pointed by (current IP + offset). The jump will occur if all channels are branched away. UIP should reference the instruction where all channels are expected to come together. JIP should reference the end of the innermost conditional block.
In GEN binary, JIP and UIP use locations src1 and src0 respectively when immediate and location src0 when reg64, where reg64 is accessed as paired DWord (regioning being <2;2,1>). dst must be IP. When the offsets are immediate, src0 regfile must be immediate.
Format: [(pred)] brc (exec_size) JIP UIP

Restriction
A brc instruction cannot use the Switch instruction option.

Syntax
[(pred)] brc (exec_size) imm32 imm32 [(pred)] brc (exec_size) reg64

Pseudocode
Evaluate(WrEn); for (n = 0; n < 32; n++) { if (WrEn[n]) { PcIP[n] = IP + UIP; } else { PcIP[n] = IP + 1; } } if (all PcIP != IP + 1) { // for all channels Jump(IP + JIP); }

Errata	Description
	A brc instruction must not be followed by any instruction requiring register indirect access on source operands.

Predication	Conditional Modifier	Saturation	Source Modifier	Src Types
Y	N	N	N	D

DWord	Bit	Description
0..3	127:96	**JIP** Project: CHV, BSW Format: S31 The byte-aligned jump distance if a jump is taken for the channel.
	95:64	**UIP** Project: CHV, BSW Format: S31 The byte aligned jump distance if a jump is taken for the instruction.

brc - Branch Converging		
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Branch Diverging

brd - Branch Diverging

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

Description
The brd instruction redirects the execution forward or backward to the instruction pointed by (current IP + offset). The jump will occur if any channels are branched away.
In GEN binary, JIP is at location src1 when immediate and at location src0 when reg32, where reg32 is accessed as a scalar DWord. The ip register must be used (for example, by the assembler) as dst.
Format: [(pred)] brd (exec_size) JIP

Restriction	Project
A brd instruction cannot use the Switch instruction option.	CHV, BSW

Syntax	Project
[(pred)] brd (exec_size) imm32 [(pred)] brd (exec_size) reg32	CHV, BSW

Pseudocode
Evaluate(WrEn); for (n = 0; n < 32; n++) { if (WrEn[n]) { PcIP[n] = IP + JIP; } else { PcIP[n] = IP + 1; } } if (any PcIP == ExIP + JIP) { // any channel Jump(ExIP + JIP); }

Errata	Description	Project
	A brd instruction must not be followed by any instruction requiring register indirect access on source operands.	CHV, BSW

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

Src Types
D

DWord	Bit	Description
0..3	127:96	**JIP** Project: CHV, BSW Format: S31 Jump Target Offset. The relative offset in bytes if a jump is taken for the instruction.
	95	**Source 0 Address Immediate [9] Sign Bit** Project: CHV, BSW

brd - Branch Diverging		
	94:91	**Src1.SrcType** Project: CHV, BSW Format: SrcType [CHV, BSW]
	90:89	**Src1.RegFile** Project: CHV, BSW Format: RegFile [CHV, BSW]
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align16') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN16 [CHV, BSW]
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN1 [CHV, BSW]
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Break

break - Break

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

Description
The break instruction is used to early-out from the inner most loop, or early out from the inner most switch block. When used in a loop, upon execution, the break instruction terminates the loop for all execution channels enabled. If all the enabled channels hit the break instruction, jump to the instruction referenced by JIP. JIP should be the offset to the end of the inner most conditional or loop block, UIP should be the offset to the while instruction of the loop block. If SPF is ON, the UIP must be used to update IP; JIP is not used in this case
The following table describes the two 32-bit instruction pointer offsets. Both the JIP and UIP are signed 32-bit numbers, added to IP pre-increment. In GEN binary, JIP and UIP are at locations src0 and src1 and must be of type DW (signed DWord integer). When the offsets are immediate, src0 regfile must be immediate.
Format: [(pred)] break (exec_size) JIP UIP

Restriction
The execution size of break instruction must be same as execution size of the corresponding while instruction.
The execution size of break instruction must be same as execution size of the corresponding while instruction.

Syntax	Project
[(pred)] break (exec_size) imm16 imm16	CHV, BSW

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.channel[n]) { PcIP[n] = IP + UIP; else { PcIP[n] = IP + 1; } } if (PcIP != (IP + 1)) { // all channels Jump(IP + JIP); }

Errata	Description	Project
	A break instruction must not be followed by any instruction requiring register indirect access on source operands.	CHV, BSW

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

DWord	Bit	Description
0..3	127:96	**JIP** Project: CHV, BSW Format: S31 The byte-aligned jump distance if a jump is taken for the channel.

break - Break		
	95:64	**UIP** Project: CHV, BSW Format: S31 The byte aligned jump distance if a jump is taken for the instruction.
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Byte Scattered Read MSD

MSD0R_BS - Byte Scattered Read MSD

Project:	CHV, BSW
Source:	DataPort 0
Length Bias:	1
Family:	Scattered R/W
Group:	Byte Scattered R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: Enable If set, indicates that the message includes the header.
	18	**Legacy Message** Default Value: 0h Project: All Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 04h Project: All Format: Opcode Byte Scattered Read message
	13	**Reserved** Project: CHV, BSW Format: MBZ Ignored
	12	**Reserved** Project: All Format: MBZ Ignored
	11:10	**Data Elements** Project: All Format: MDC_DS [CHV, BSW] Specifies the number of Bytes to be read or written per Dword

<table>
<tr><th colspan="3">MSD0R_BS - Byte Scattered Read MSD</th></tr>
<tr><td rowspan="3"></td><td>9</td><td>Reserved<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table>Ignored</td></tr>
<tr><td>8</td><td>SIMD Mode<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MDC_SM2 [CHV, BSW]</td></tr></table>Specifies the SIMD mode of the message (number of slots processed)</td></tr>
<tr><td>7:0</td><td>Binding Table Index<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MDC_BTS_SLM_A32 [CHV, BSW]</td></tr></table>Specifies the Binding Table Index for the message</td></tr>
</table>

Byte Scattered Write MSD

MSD0W_BS - Byte Scattered Write MSD

Project:	CHV, BSW
Source:	DataPort 0
Length Bias:	1
Family:	Scattered R/W
Group:	Byte Scattered R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: Enable If set, indicates that the message includes the header.
	18	**Legacy Message** Default Value: 0h Project: All Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 0Ch Project: All Format: Opcode Byte Scattered Write message
	13:12	**Reserved** Project: All Format: MBZ Ignored
	11:10	**Data Elements** Project: All Format: MDC_DS [CHV, BSW] Specifies the number of Bytes to be read or written per Dword
	9	**Reserved** Project: All Format: MBZ Ignored

MSD0W_BS - Byte Scattered Write MSD		
	8	**SIMD Mode** Project: All Format: MDC_SM2 [CHV, BSW] Specifies the SIMD mode of the message (number of slots processed)
	7:0	**Binding Table Index** Project: All Format: MDC_BTS_SLM_A32 [CHV, BSW] Specifies the Binding Table Index for the message

Call

call - Call

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

Description
The call instruction jumps to a subroutine. It can be predicated or non-predicated. If non-predicated, all enabled channels jump to the subroutine. If predicated, only the channels enabled by PMask jump to the subroutine; the rest of the channels move to the next instruction after the call instruction. If none of the channels jump into the subroutine, the call instruction is treated as a nop. In case of a jump, the call instruction stores the return IP onto the first DWord of the destination register and stores the CallMask in the second DWord of the destination register. When SPF is on, the predication control must be scalar.
The following table describes JIP, the jump offset. JIP can be an immediate or register value. When a jump occurs, this value is added to IP pre-increment. For CHV, BSW, in GEN binary, JIP is at location src1 when immediate and at location src0 when in a register. The IP register must be put (for example, by the assembler) at dst location. When offsets are immediate, src0 must be null.
Format: [(pred)] call (exec_size) dst JIP

Restriction
The call instruction must have DWord source and destination type, and the destination must be QWord aligned.
A call instruction must use a Switch
A call instruction must be followed by an instruction that supports Switch. When call takes a jump, the first instruction must have a Switch.

Syntax
[(pred)] call (exec_size) reg imm32 [(pred)] call (exec_size) reg reg32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { PcIP[n] = IP + JIP; CallMask[n] = 1; } else { PcIP[n] = IP + 1; CallMask[n] = 0; } } if (PcIP[n] != (IP + 1)) { // any channel jumped dst.chan[0] = IP + 1; dst.chan[1] = CallMask; Jump(IP + JIP); }

Errata	Description
	A call instruction must not be followed by any instruction requiring register indirect access on source operands.

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

Dst Types
D, UD

DWord	Bit	Description
0..3	127:96	**JIP** Project: CHV, BSW Format: S31 Jump Target Offset. The relative offset in bytes if a jump is taken for the instruction.
	95	**Source 0 Address Immediate [9] Sign Bit** Project: CHV, BSW
	94:91	**Src1.SrcType** Project: CHV, BSW Format: SrcType [CHV, BSW]
	90:89	**Src1.RegFile** Project: CHV, BSW Format: RegFile [CHV, BSW]
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align16') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN16 [CHV, BSW]
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN1 [CHV, BSW]
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Call Absolute

calla - Call Absolute

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The calla instruction jumps to a subroutine. It can be predicated or non-predicated. If non-predicated, all enabled channels jump to the subroutine. If predicated, only the channels enabled by PMask jump to the subroutine; the rest of the channels move to the next instruction after the calla instruction. If none of the channels jump into the subroutine, the calla instruction is treated as a nop. In case of a jump, the call instruction stores the return IP onto the first DWord of the destination register and stores the CallMask in the second DWord of the destination register. If SPF is ON, none of the PcIP are updated. When SPF is on, the predication control must be scalar. The difference between calla and call is that calla uses JIP as the IP value rather than adding it to the IP value.

Format: [(pred)] calla (exec_size) dst JIP

Restriction
The calla instruction must have DWord source and destination type, and the destination must be QWord-aligned.
The src0 regioning control must be <2;2,1>

Syntax	Project
[(pred)] calla (exec_size) reg imm32	CHV, BSW

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.channel[n]) { PcIP[n] = JIP; CallMask[n] = 1; } else { PcIP[n] = IP + 1; CallMask[n] = 0; } } if (PcIP[n] != (IP + 1)) { // any channel jumped dst.chan[0] = IP + 1; dst.chan[1] = CallMask; Jump(JIP); }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

Dst Types
D, UD

DWord	Bit	Description
0..3	127:96	**JIP** Project: CHV, BSW Format: S31 Jump Target Offset. The relative offset in bytes if a jump is taken for the instruction.
	95	**Source 0 Address Immediate [9] Sign Bit** Project: CHV, BSW

calla - Call Absolute

	94:91	**Src1.SrcType** Project: CHV, BSW Format: SrcType [CHV, BSW]
	90:89	**Src1.RegFile** Project: CHV, BSW Format: RegFile [CHV, BSW]
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align16') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN16 [CHV, BSW]
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN1 [CHV, BSW]
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Compare

cmp - Compare

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The cmp instruction performs component-wise comparison of src0 and src1 and stores the results in the selected flag register and in dst. It takes component-wise subtraction of src0 and src1, evaluating the conditional code (excluding NS signal) based on the conditional modifier, and storing the conditional bits in bit-packed form in the destination flag register and all bits of dst channels. If the dst is not null, for the enabled channels, then all bits of the destination channel will contain the flag value for the channel. When the instruction operates on packed word format, one general register may store up to 16 such comparison results. In DWord format, one general register may store up to 8 results. A conditional modifier must be specified; the conditional modifier field cannot be 0000b. The comparison does not use the NS (NaN source) signals, as described in the Creating Conditional Flags section. Accordingly the conditional modifier should not be .u (unordered). For each enabled channel 0b or 1b is assigned to the appropriate flag bit and 0/all zeros or all ones (e.g, byte 0xFF, word 0xFFFF, DWord 0xFFFFFFFF) is assigned to dst. When any source type is floating-point, the cmp instruction obeys the rules described in the tables in the Floating Point Modes section of the Data Types chapter.

Format: [(pred)] cmp[.cmod] (exec_size) dst src0 src1

Restriction
Accumulator cannot be destination, implicit or explicit. The destination must be a general register or the null register.

Syntax
[(pred)] cmp[.cmod] (exec_size) reg reg reg [(pred)] cmp[.cmod] (exec_size) reg reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { bitMask[n] = 0; if (WrEn.chan[n]) { results[n] = src0.chan[n] - src1.chan[n]; bitMask[n] = Condition(results[n]); dst.chan[n] = bitMask[n]; // All bits for dst channel } } flag# = bitMask;

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	N	Y

Src Types	Dst Types	Project
*B,*W,*D	*B,*W,*D	
*B,*W,*D	F	
F	F	
DF	DF	CHV, BSW
HF	HF	CHV, BSW

cmp - Compare

*B,*W,*D	HF	CHV, BSW
*W,*D,*Q	*W,*D,*Q	CHV, BSW
HF, F	HF, F	CHV, BSW

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Compare NaN

cmpn - Compare NaN

Project:	CHV, BSW
Source:	Eulsa
Length Bias:	4

The cmpn instruction performs component-wise special-NaN comparison of src0 and src1 and stores the results in the selected flag register and in dst. It takes component-wise subtraction of src0 and src1, evaluating the conditional signals including NS based on the conditional modifier, and storing the conditional flag bits in bit-packed form in the destination flag register and all bits of dst channels. If the dst is not null, for the enabled channels, then all bits of the destination channel will contain the flag value for the channel. When the instruction operates on packed word format, one general register may store up to 16 such comparison results. In DWord format, one general register may store up to 8 results. A conditional modifier must be specified; the conditional modifier field cannot be 0000b. More information about the conditional signals used is in the Creating Conditional Flags section. For each enabled channel 0b or 1b is assigned to the appropriate flag bit and 0/all zeros or all ones (e.g, byte 0xFF, word 0xFFFF, DWord 0xFFFFFFFF) is assigned to dst. Min/Max instructions use cmpn to select the destination from the input sources (see the Min Max of Floating Point Numbers section for details).

Format: [(pred)] cmpn[.cmod] (exec_size) dst src0 src1

Restriction
Accumulator cannot be destination, implicit or explicit. The destination must be a general register or the null register.
.l and .ge are the only two conditional modifiers are supported for this instruction.

Syntax
[(pred)] cmpn[.cmod] (exec_size) reg reg reg [(pred)] cmpn[.cmod] (exec_size) reg reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { bitMask[n] = 0; if (WrEn.chan[n]) { results[n] = src0.chan[n] - src1.chan[n]; bitMask[n] = ConditionNaN(results[n]); dst.chan[n][0] = bitMask[n]; // All bits for dst channel } } flag# = bitMask;

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	N	Y

Src Types	Dst Types	Project
*B,*W,*D	*B,*W,*D	
*B,*W,*D	F	
F	F	
DF	DF	CHV, BSW
HF	HF	CHV, BSW

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Conditional Select

csel - Conditional Select

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The csel instruction selectively moves components in src0 or src1 to the dst based on the result of the compare of src2 with zero. If the channel condition is true, data in src0 is moved into dst. Otherwise, data in src1 is moved into dst. The csel instruction provides the function of a cmp followed by sel. The instruction must not be used if cmpn is required. The instruction does not update the flag register.
The comparison follows the same rule as cmp instruction for that data type.

Format: csel (exec_size) dst src0 src1 src2

Syntax
csel[.cmod] (exec_size) reg reg reg

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { bitMask[n] = 0; if (EMask.chan[n]) { result[n] = src2.chan[n] - 0; bitMask[n] = Condition(result[n]); if (bitMask[n] = 1) { dst.chan[n] = src0.chan[n]; } else { dst.chan[n] = src1.chan[n]; } } }

Predication	Conditional Modifier	Saturation	Source Modifier
N	Y	Y	Y

Src Types	Dst Types
F	F

DWord	Bit	Description
0..3	127:126	**Reserved** Format: MBZ
	125:106	**Source 2** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC [CHV, BSW]
	105	**Reserved** Format: MBZ
	104:85	**Source 1** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC [CHV, BSW]
	84	**Reserved** Format: MBZ
	83:64	**Source 0** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC [CHV, BSW]

csel - Conditional Select

	63:56	**Destination Register Number** Format: DstRegNum [CHV, BSW]
	55:53	**Destination Subregister Number** Format: DstSubRegNum[2:0]
	52:49	**Destination Channel Enable** Format: ChanEn[4] Four channel enables are defined for controlling which channels are written into the destination region. These channel mask bits are applied in a modulo-four manner to all ExecSize channels. There is 1-bit Channel Enable for each channel within the group of 4. If the bit is cleared, the write for the corresponding channel is disabled. If the bit is set, the write is enabled. Mnemonics for the bit being set for the group of 4 are x, y, z, and w, respectively, where x corresponds to Channel 0 in the group and w corresponds to channel 3 in the group
	48:42	**Reserved** Format: MBZ
	41:40	**Source 2 Modifier** Exists If: //([Property[Source Modifier]=='true') Format: SrcMod [CHV, BSW]
	39:38	**Source 1 Modifier** Exists If: //([Property[Source Modifier]=='true') Format: SrcMod [CHV, BSW]
	41:36	**Reserved** Exists If: //([Property[Source Modifier]=='false') Format: MBZ
	37:36	**Source 0 Modifier** Exists If: //([Property[Source Modifier]=='true') Format: SrcMod [CHV, BSW]
	35	**Reserved** Format: MBZ
	34	**Reserved** Format: MBZ
	33	**Flag Subregister Number** This field contains the flag subregister number for instructions with a non-zero Conditional Modifier.
	32	**Reserved** Format: MBZ
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Conditional Send Message

sendc - Conditional Send Message

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The sendc instruction has the same behavior as the send instruction except the following. sendc first checks the dependent threads inside the Thread Dependency Register. There are up to 8 dependent threads in the TDR register. The sendc instruction executes only when all the dependent threads in the TDR register are retired. Wait for dependencies in the TDR Register to clear, then send a message stored in registers starting at src to a shared function identified by exdesc along with control from desc with a general register writeback location at dst.

Format: [(pred)] sendc (exec_size) dst src0 exdesc desc

Restriction
The sendc instruction has the same restrictions as the send instruction.

Pseudocode
if (TDR[7] ... || TDR[2] || TDR[1] || TDR[0]) { wait; } Evaluate(WrEn); MsgChEnable = WrEn; SourceReg = src0.RegNum; MessageEnqueue(MsgChEnable, ResponseReg, SourceReg, desc, exdesc);

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

DWord	Bit	Description
0..3	127:96	**Message** Format: EU_INSTRUCTION_OPERAND_SEND_MSG [CHV, BSW]
	95:89	**Flags** Format: EU_INSTRUCTION_FLAGS
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN1 [CHV, BSW]
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align16') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN16 [CHV, BSW]
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:28	**Controls B** Format: EU_INSTRUCTION_CONTROLS_B [CHV, BSW]

sendc - Conditional Send Message		
	27:24	**Shared Function ID (SFID)** Format: SFID [CHV, BSW]
	23:8	**Controls A** Format: EU_INSTRUCTION_CONTROLS_A [CHV, BSW]
	7	**Reserved** Format: MBZ
	6:0	**Opcode** Format: EU_OPCODE [CHV, BSW]

Constant Cache Dword Scattered Read MSD

MSD_CC_DWS - Constant Cache Dword Scattered Read MSD

Project:	CHV, BSW
Source:	Read-Only DataPort
Length Bias:	1
Family:	Scattered R/W
Group:	DW Scattered R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: Enable If set, indicates that the message includes the header.
	18	**Legacy Message** Default Value: 0h Project: All Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 03h Project: All Format: Opcode Dword Scattered Read message
	13	**Invalidate After Read** Project: All Format: MDC_IAR [CHV, BSW] Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12	**Reserved** Project: All Format: MBZ Ignored
	11:10	**Reserved** Project: CHV, BSW Format: MBZ

MSD_CC_DWS - Constant Cache Dword Scattered Read MSD

	9	**Legacy SIMD Mode** Default Value: 1h Project: All Format: Opcode Must be set for compatibility.
	8	**SIMD Mode** Project: All Format: MDC_SM2 [CHV, BSW] Specifies the SIMD mode of the message (number of slots processed)
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Constant Cache Oword Block Read MSD

MSD_CC_OWB - Constant Cache Oword Block Read MSD

Project:	CHV, BSW
Source:	Read-Only DataPort
Length Bias:	1
Family:	Block R/W
Group:	OW Block R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header.
	18	**Reserved** Project: All Format: MBZ Ignored
	17:14	**Message Type** Default Value: 00h Project: All Format: Opcode Oword Block Read Constant Cache message
	13	**Invalidate After Read** Project: All Format: MDC_IAR [CHV, BSW] Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12:11	**Reserved** Project: All Format: MBZ Ignored
	10:8	**Data Elements** Project: All Format: MDC_DB_OW [CHV, BSW] Specifies the number of contiguous Owords to be read or written

MSD_CC_OWB - Constant Cache Oword Block Read MSD		
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Constant Cache Oword Dual Block Read MSD

MSD_CC_OWDB - Constant Cache Oword Dual Block Read MSD

Project:	CHV, BSW
Source:	Read-Only DataPort
Length Bias:	1
Family:	Block R/W
Group:	OW Dual Block R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: Enable If set, indicates that the message includes the header.
	18	**Reserved** Project: All Format: MBZ Ignored
	17:14	**Message Type** Default Value: 02h Project: All Format: Opcode Oword Block Read message
	13	**Invalidate After Read** Project: All Format: MDC_IAR [CHV, BSW] Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12:10	**Reserved** Project: All Format: MBZ Ignored
	9:8	**Data Elements** Project: All Format: MDC_DB_OWD [CHV, BSW] Specifies the number of contiguous Owords to be read or written

MSD_CC_OWDB - Constant Cache Oword Dual Block Read MSD		
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Constant Cache Oword Unaligned Block Read MSD

MSD_CC_OWUB - Constant Cache Oword Unaligned Block Read MSD

Project:	CHV, BSW
Source:	Read-Only DataPort
Length Bias:	1
Family:	Block R/W
Group:	OW Unaligned Block R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header.
	18	**Reserved** Project: All Format: MBZ Ignored
	17:14	**Message Type** Default Value: 01h Project: All Format: Opcode Oword Unaligned Block Read Constant Cache message
	13:11	**Reserved** Project: All Format: MBZ Ignored
	10:8	**Data Elements** Project: All Format: MDC_DB_OW [CHV, BSW] Specifies the number of contiguous Owords to be read
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Continue

cont - Continue

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

Description
The cont instruction disables execution for the subset of channels for the remainder of the current loop iteration. Channels remain disabled until right before the while instuction or right before the condition check code block for the while instruction. If all enabled channels hit this instruction, jump to the instruction referenced by JIP where execution continues. UIP should always reference the loop's associated while instruction. JIP should point to the last instruction of the inner most conditional block if the cont instruction is inside a conditional block. In case of the break instruction directly under the loop, the JIP and the UIP are the same. If SPF is ON, the UIP must be used to update IP; JIP is not used in this case.
The following table describes the two 32-bit instruction pointer offsets. Both the JIP and UIP are signed 32-bit numbers, added to IP pre-increment. In GEN binary, JIP and UIP are at locations src0 and src1 and must be of type DW (signed DWord integer). When the offsets are immediate, src0 regfile must be immediate.
Format: [(pred)] cont (exec_size) JIP UIP

Restriction
The execution size must be the same for the while, break, and cont instructions of the same code block.

Syntax
[(pred)] cont (exec_size) imm32 imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.channel[n]) { if (PMask[n]) { // PMask is for all channels enabled for the cont instruction. PcIP[n] = IP + UIP; } else { PcIP[n] = IP + 1; } } } for (n = exec_size; n < 32; n++) { PcIP[n] = IP + 1; } if (PcIP != (IP + 1)) { // all channels true Jump(IP + JIP); }

Errata	Description
	A cont instruction must not be followed by any instruction requiring register indirect access on source operands.

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

DWord	Bit	Description
0..3	127:96	**JIP** Project: CHV, BSW Format: S31 The byte-aligned jump distance if a jump is taken for the channel.

cont - Continue

	95:64	**UIP** Project: CHV, BSW Format: S31 The byte aligned jump distance if a jump is taken for the instruction.
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Count Bits Set

cbit - Count Bits Set

Project:	CHV, BSW
Source:	Eulsa
Length Bias:	4

The cbit instruction counts component-wise the total bits set in src0 and stores the resulting counts in dst.

Format: [(pred)] cbit (exec_size) dst src0

Restriction
No accumulator access, implicit or explicit.

Syntax
[(pred)] cbit (exec_size) reg reg [(pred)] cbit (exec_size) reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { UD cnt = 0; UD val = src0.chan[n]; while (val) { if (val & 1) { cnt ++; } val = val » 1; } dst.chan[n] = cnt; } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

Src Types	Dst Types
UB, UW, UD	UD

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32 [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Dot Product 2

dp2 - Dot Product 2

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The dp2 instruction performs a two-wide dot product on four-tuple vector basis and storing the same scalar result per four tuple to all four channels in dst. This instruction is similar to dp4 except that every third and fourth element of src0 (post-source-swizzle if present) are not involved in the computation. The dot product of two vectors of equal length is the sum of the products of each pair of corresponding elements. The dp4 instruction includes all four elements of each vector in the dot product. The dp3 instruction includes the first three elements of each vector in the dot product.

Format: [(pred)] dp2[.cmod] (exec_size) dst src0 src1

Restriction
Execution size cannot be less than 4.
Horizontal strides must be 1.
Source operands cannot be accumulators.

Syntax
[(pred)] dp2[.cmod] (exec_size) reg reg reg [(pred)] dp2[.cmod] (exec_size) reg reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n += 4) { fTmp = src0.chan[n] * src1.chan[n] + src0.chan[n+1] * src1.chan[n+1]; if (WrEn.chan[n]) dst.chan[n] = fTmp; if (WrEn.chan[n+1]) dst.chan[n+1] = fTmp; if (WrEn.chan[n+2]) dst.chan[n+2] = fTmp; if (WrEn.chan[n+3]) dst.chan[n+3] = fTmp; }

Predication	Conditional Modifier	Saturation	Source Modifier	Src Types	Dst Types
Y	Y	Y	Y	F	F

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Dot Product 3

dp3 - Dot Product 3

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The dp3 instruction performs a three-wide dot product on four-tuple vector basis and storing the same scalar result per four tuple to all four channels in dst. This instruction is similar to dp4 except that every fourth element of src0 (post-source-swizzle if present) is not involved in the computation. The dot product of two vectors of equal length is the sum of the products of each pair of corresponding elements. The dp4 instruction includes all four elements of each vector in the dot product. The dp2 instruction includes the first two elements of each vector in the dot product.

Format: [(pred)] dp3[.cmod] (exec_size) dst src0 src1

Restriction
Execution size cannot be less than 4.
Horizontal strides must be 1.
Source operands cannot be accumulators.

Syntax
[(pred)] dp3[.cmod] (exec_size) reg reg reg [(pred)] dp3[.cmod] (exec_size) reg reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n += 4) { fTmp = src0.chan[n] * src1.chan[n] + src0.chan[n+1] * src1.chan[n+1] + src0.chan[n+2] * src1.chan[n+2]; if (WrEn.chan[n]) dst.chan[n] = fTmp; if (WrEn.chan[n+1]) dst.chan[n+1] = fTmp; if (WrEn.chan[n+2]) dst.chan[n+2] = fTmp; if (WrEn.chan[n+3]) dst.chan[n+3] = fTmp; }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types
F	F

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM [CHV, BSW]

dp3 - Dot Product 3		
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Dot Product 4

dp4 - Dot Product 4

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The dp4 instruction performs a four-wide dot product on four-tuple vector basis and storing the same scalar result per four tuple to all four channels in dst. The dot product of two vectors of equal length is the sum of the products of each pair of corresponding elements.

Format: [(pred)] dp4[.cmod] (exec_size) dst src0 src1

Restriction
Execution size cannot be less than 4.
Horizontal strides must be 1.
Source operands cannot be accumulators.

Syntax
[(pred)] dp4[.cmod] (exec_size) reg reg reg [(pred)] dp4[.cmod] (exec_size) reg reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n += 4) { fTmp = src0.chan[n] * src1.chan[n] + src0.chan[n+1] * src1.chan[n+1] + src0.chan[n+2] * src1.chan[n+2] + src0.chan[n+3] * src1.chan[n+3]; if (WrEn.chan[n]) dst.chan[n] = fTmp; if (WrEn.chan[n+1]) dst.chan[n+1] = fTmp; if (WrEn.chan[n+2]) dst.chan[n+2] = fTmp; if (WrEn.chan[n+3]) dst.chan[n+3] = fTmp; }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types
F	F

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Dot Product Homogeneous

dph - Dot Product Homogeneous

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The dph instruction performs a four-wide homogeneous dot product on four-tuple vector basis and storing the same scalar result per four tuple to all four channels in dst. This instruction is similar to dp4 except that every fourth element of src0 (post-source-swizzle if present) is forced to 1.0f. Use the dp4 instruction to do a four-wide dot product that includes all elements of src0 and src1.

Format: [(pred)] dph[.cmod] (exec_size) dst src0 src1

Restriction
Execution size cannot be less than 4.
Horizontal strides must be 1.
Source operands cannot be accumulators.

Syntax
[(pred)] dph[.cmod] (exec_size) reg reg reg [(pred)] dph[.cmod] (exec_size) reg reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n += 4) { fTmp = src0.chan[n] * src1.chan[n] + src0.chan[n+1] * src1.chan[n+1] + src0.chan[n+2] * src1.chan[n+2] + src1.chan[n+3]; // Use 1.0f in place of src0.chan[n+3]. if (WrEn.chan[n]) dst.chan[n] = fTmp; if (WrEn.chan[n+1]) dst.chan[n+1] = fTmp; if (WrEn.chan[n+2]) dst.chan[n+2] = fTmp; if (WrEn.chan[n+3]) dst.chan[n+3] = fTmp; }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types
F	F

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Dword Atomic Counter Binary with Return Data Operation MSD

MSD1R_DWAC2 - Dword Atomic Counter Binary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Atomic Counter Binary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header
	18:14	**Message Type** Default Value: 0Bh Project: All Format: Opcode Atomic Counter Operation message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**SIMD Mode** Project: All Format: MDC_SM2RS [CHV, BSW] Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP2 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Dword Atomic Counter Binary Write Only Operation MSD

MSD1W_DWAC2 - Dword Atomic Counter Binary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Atomic Counter Binary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header
	18:14	**Message Type** Default Value: 0Bh Project: All Format: Opcode Atomic Counter Operation message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**SIMD Mode** Project: All Format: MDC_SM2RS [CHV, BSW] Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP2 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Dword Atomic Counter Unary with Return Data Operation MSD

MSD1R_DWAC1 - Dword Atomic Counter Unary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Atomic Counter Unary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header
	18:14	**Message Type** Default Value: 0Bh Project: All Format: Opcode Atomic Counter Operation message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**SIMD Mode** Project: All Format: MDC_SM2RS [CHV, BSW] Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP1 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Dword Atomic Counter Unary Write Only Operation MSD

MSD1W_DWAC1 - Dword Atomic Counter Unary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Atomic Counter Unary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header
	18:14	**Message Type** Default Value: 0Bh Project: All Format: Opcode Atomic Counter Operation message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**SIMD Mode** Project: All Format: MDC_SM2RS [CHV, BSW] Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP1 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Dword Scattered Read MSD

MSD0R_DWS - Dword Scattered Read MSD

Project:	CHV, BSW
Source:	DataPort 0
Length Bias:	1
Family:	Scattered R/W
Group:	DW Scattered R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: Enable If set, indicates that the message includes the header.
	18	**Legacy Message** Default Value: 0h Project: All Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 03h Project: All Format: Opcode Dword Scattered Read message
	13	**Invalidate After Read** Project: All Format: MDC_IAR [CHV, BSW] Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12	**Reserved** Project: All Format: MBZ Ignored
	11:10	**Reserved** Project: CHV, BSW Format: MBZ Ignored

MSD0R_DWS - Dword Scattered Read MSD

	9	**Legacy SIMD Mode** Default Value: 1h Project: All Format: Opcode Must be set for compatibility.
	8	**SIMD Mode** Project: All Format: MDC_SM2 [CHV, BSW] Specifies the SIMD mode of the message (number of slots processed)
	7:0	**Binding Table Index** Project: CHV, BSW Format: MDC_BTS_A32 [CHV, BSW] Specifies the Binding Table Index for the message

Dword Scattered Write MSD

MSD0W_DWS - Dword Scattered Write MSD

Project:	CHV, BSW
Source:	DataPort 0
Length Bias:	1
Family:	Scattered R/W
Group:	DW Scattered R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: Enable If set, indicates that the message includes the header.
	18	**Legacy Message** Default Value: 0h Project: All Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 0Bh Project: All Format: Opcode Dword Scattered Write message
	13:12	**Reserved** Project: All Format: MBZ Ignored
	11:10	**Reserved** Project: CHV, BSW Format: MBZ Ignored
	9	**Legacy SIMD Mode** Default Value: 1h Project: All Format: Opcode Must be set for compatibility.

MSD0W_DWS - Dword Scattered Write MSD

	8	**SIMD Mode** Project: All Format: MDC_SM2 [CHV, BSW] Specifies the SIMD mode of the message (number of slots processed)
	7:0	**Binding Table Index** Project: CHV, BSW Format: MDC_BTS_A32 [CHV, BSW] Specifies the Binding Table Index for the message

Dword SIMD4x2 Atomic Counter Binary with Return Data Operation MSD

MSD1R_DWAC2_4x2 - Dword SIMD4x2 Atomic Counter Binary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Atomic Counter Binary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header
	18:14	**Message Type** Default Value: 0Ch Project: All Format: Opcode Atomic Counter Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**Reserved**
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP2 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Dword SIMD4x2 Atomic Counter Binary Write Only Operation MSD

MSD1W_DWAC2_4x2 - Dword SIMD4x2 Atomic Counter Binary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Atomic Counter Binary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header
	18:14	**Message Type** Default Value: 0Ch Project: All Format: Opcode Atomic Counter Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Reserved**
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP2 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Dword SIMD4x2 Atomic Counter Unary with Return Data Operation MSD

MSD1R_DWAC1_4x2 - Dword SIMD4x2 Atomic Counter Unary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Atomic Counter Unary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header
	18:14	**Message Type** Default Value: 0Ch Project: All Format: Opcode Atomic Counter Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**Reserved**
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP1 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Dword SIMD4x2 Atomic Counter Unary Write Only Operation MSD

MSD1W_DWAC1_4x2 - Dword SIMD4x2 Atomic Counter Unary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Atomic Counter Unary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header
	18:14	**Message Type** Default Value: 0Ch Project: All Format: Opcode Atomic Counter Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Reserved**
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP1 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Dword SIMD4x2 Typed Atomic Integer Binary with Return Data Operation MSD

MSD1R_DWTAI2_4x2 - Dword SIMD4x2 Typed Atomic Integer Binary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Typed Atomic Operation
Group:	Dword Typed Atomic Integer Binary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 07h Project: All Format: Opcode Typed Atomic Integer Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**Reserved**
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP2 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Dword SIMD4x2 Typed Atomic Integer Binary Write Only Operation MSD

MSD1W_DWTAI2_4x2 - Dword SIMD4x2 Typed Atomic Integer Binary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Typed Atomic Operation
Group:	Dword Typed Atomic Integer Binary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 07h Project: All Format: Opcode Typed Atomic Integer Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Reserved**
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP2 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Dword SIMD4x2 Typed Atomic Integer Trinary with Return Data Operation MSD

MSD1R_DWTAI3_4x2 - Dword SIMD4x2 Typed Atomic Integer Trinary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Typed Atomic Operation
Group:	Dword Typed Atomic Integer Trinary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 07h Project: All Format: Opcode Typed Atomic Integer Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**Reserved**
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP3S [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Dword SIMD4x2 Typed Atomic Integer Trinary Write Only Operation MSD

MSD1W_DWTAI3_4x2 - Dword SIMD4x2 Typed Atomic Integer Trinary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Typed Atomic Operation
Group:	Dword Typed Atomic Integer Trinary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 07h Project: All Format: Opcode Typed Atomic Integer Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Reserved**
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP3S [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Dword SIMD4x2 Typed Atomic Integer Unary with Return Data Operation MSD

MSD1R_DWTAI1_4x2 - Dword SIMD4x2 Typed Atomic Integer Unary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Typed Atomic Operation
Group:	Dword Typed Atomic Integer Unary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 07h Project: All Format: Opcode Typed Atomic Integer Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**Reserved**
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP1 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Dword SIMD4x2 Typed Atomic Integer Unary Write Only Operation MSD

MSD1W_DWTAI1_4x2 - Dword SIMD4x2 Typed Atomic Integer Unary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Typed Atomic Operation
Group:	Dword Typed Atomic Integer Unary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 07h Project: All Format: Opcode Typed Atomic Integer Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Reserved**
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP1 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Dword SIMD4x2 Untyped Atomic Integer Binary with Return Data Operation MSD

MSD1R_DWAI2_4x2 - Dword SIMD4x2 Untyped Atomic Integer Binary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Untyped Atomic Integer Binary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 03h Project: All Format: Opcode Untyped Atomic Integer Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**Reserved**
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP2 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS_SLM_A32 [CHV, BSW] Specifies the Binding Table Index for the message

Dword SIMD4x2 Untyped Atomic Integer Binary Write Only Operation MSD

MSD1W_DWAI2_4x2 - Dword SIMD4x2 Untyped Atomic Integer Binary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Untyped Atomic Integer Binary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 03h Project: All Format: Opcode Untyped Atomic Integer Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Reserved**
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP2 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS_SLM_A32 [CHV, BSW] Specifies the Binding Table Index for the message

Dword SIMD4x2 Untyped Atomic Integer Trinary with Return Data Operation MSD

MSD1R_DWAI3_4x2 - Dword SIMD4x2 Untyped Atomic Integer Trinary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Untyped Atomic Integer Trinary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 03h Project: All Format: Opcode Untyped Atomic Integer Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**Reserved**
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP3 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS_SLM_A32 [CHV, BSW] Specifies the Binding Table Index for the message

Dword SIMD4x2 Untyped Atomic Integer Trinary Write Only Operation MSD

MSD1W_DWAI3_4x2 - Dword SIMD4x2 Untyped Atomic Integer Trinary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Untyped Atomic Integer Trinary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 03h Project: All Format: Opcode Untyped Atomic Integer Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Reserved**
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP3 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS_SLM_A32 [CHV, BSW] Specifies the Binding Table Index for the message

Dword SIMD4x2 Untyped Atomic Integer Unary with Return Data Operation MSD

MSD1R_DWAI1_4x2 - Dword SIMD4x2 Untyped Atomic Integer Unary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Untyped Atomic Integer Unary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 03h Project: All Format: Opcode Untyped Atomic Integer Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**Reserved**
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP1 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS_SLM_A32 [CHV, BSW] Specifies the Binding Table Index for the message

Dword SIMD4x2 Untyped Atomic Integer Unary Write Only Operation MSD

MSD1W_DWAI1_4x2 - Dword SIMD4x2 Untyped Atomic Integer Unary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Untyped Atomic Integer Unary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 03h Project: All Format: Opcode Untyped Atomic Integer Operation SIMD4x2 message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Reserved**
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP1 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS_SLM_A32 [CHV, BSW] Specifies the Binding Table Index for the message

Dword Typed Atomic Integer Binary with Return Data Operation MSD

MSD1R_DWTAI2 - Dword Typed Atomic Integer Binary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Typed Atomic Operation
Group:	Dword Typed Atomic Integer Binary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 06h Project: All Format: Opcode Typed Atomic Integer Operation message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**Slot Group** Project: All Format: MDC_SG2 [CHV, BSW] Specifies the Slot Group mode of the message (which slots are processed)
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP2 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Dword Typed Atomic Integer Binary Write Only Operation MSD

MSD1W_DWTAI2 - Dword Typed Atomic Integer Binary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Typed Atomic Operation
Group:	Dword Typed Atomic Integer Binary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 06h Project: All Format: Opcode Typed Atomic Integer Operation message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Slot Group** Project: All Format: MDC_SG2 [CHV, BSW] Specifies the Slot Group mode of the message (which slots are processed)
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP2 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Dword Typed Atomic Integer Trinary with Return Data Operation MSD

MSD1R_DWTAI3 - Dword Typed Atomic Integer Trinary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Typed Atomic Operation
Group:	Dword Typed Atomic Integer Trinary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 06h Project: All Format: Opcode Typed Atomic Integer Operation message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**Slot Group** Project: All Format: MDC_SG2 [CHV, BSW] Specifies the Slot Group mode of the message (which slots are processed)
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP3S [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Dword Typed Atomic Integer Trinary Write Only Operation MSD

MSD1W_DWTAI3 - Dword Typed Atomic Integer Trinary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Typed Atomic Operation
Group:	Dword Typed Atomic Integer Trinary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 06h Project: All Format: Opcode Typed Atomic Integer Operation message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Slot Group** Project: All Format: MDC_SG2 [CHV, BSW] Specifies the Slot Group mode of the message (which slots are processed)
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP3S [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Dword Typed Atomic Integer Unary with Return Data Operation MSD

MSD1R_DWTAI1 - Dword Typed Atomic Integer Unary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Typed Atomic Operation
Group:	Dword Typed Atomic Integer Unary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 06h Project: All Format: Opcode Typed Atomic Integer Operation message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**Slot Group** Project: All Format: MDC_SG2 [CHV, BSW] Specifies the Slot Group mode of the message (which slots are processed)
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP1 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Dword Typed Atomic Integer Unary Write Only Operation MSD

MSD1W_DWTAI1 - Dword Typed Atomic Integer Unary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Typed Atomic Operation
Group:	Dword Typed Atomic Integer Unary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 06h Project: All Format: Opcode Typed Atomic Integer Operation message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Slot Group** Project: All Format: MDC_SG2 [CHV, BSW] Specifies the Slot Group mode of the message (which slots are processed)
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP1 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Dword Untyped Atomic Integer Binary with Return Data Operation MSD

MSD1R_DWAI2 - Dword Untyped Atomic Integer Binary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Untyped Atomic Integer Binary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 02h Project: All Format: Opcode Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**SIMD Mode** Project: All Format: MDC_SM2R [CHV, BSW] Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP2 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS_SLM_A32 [CHV, BSW] Specifies the Binding Table Index for the message

Dword Untyped Atomic Integer Binary Write Only Operation MSD

MSD1W_DWAI2 - Dword Untyped Atomic Integer Binary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Untyped Atomic Integer Binary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 02h Project: All Format: Opcode Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**SIMD Mode** Project: All Format: MDC_SM2R [CHV, BSW] Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP2 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS_SLM_A32 [CHV, BSW] Specifies the Binding Table Index for the message

Dword Untyped Atomic Integer Trinary with Return Data Operation MSD

MSD1R_DWAI3 - Dword Untyped Atomic Integer Trinary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Untyped Atomic Integer Trinary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 02h Project: All Format: Opcode Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**SIMD Mode** Project: All Format: MDC_SM2R [CHV, BSW] Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP3 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS_SLM_A32 [CHV, BSW] Specifies the Binding Table Index for the message

Dword Untyped Atomic Integer Trinary Write Only Operation MSD

MSD1W_DWAI3 - Dword Untyped Atomic Integer Trinary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Untyped Atomic Integer Trinary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 02h Project: All Format: Opcode Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**SIMD Mode** Project: All Format: MDC_SM2R [CHV, BSW] Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP3 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS_SLM_A32 [CHV, BSW] Specifies the Binding Table Index for the message

Dword Untyped Atomic Integer Unary with Return Data Operation MSD

MSD1R_DWAI1 - Dword Untyped Atomic Integer Unary with Return Data Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Untyped Atomic Integer Unary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 02h Project: All Format: Opcode Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 1h Project: All Format: Opcode Specifies that return data is sent back to the thread.
	12	**SIMD Mode** Project: All Format: MDC_SM2R [CHV, BSW] Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP1 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS_SLM_A32 [CHV, BSW] Specifies the Binding Table Index for the message

Dword Untyped Atomic Integer Unary Write Only Operation MSD

MSD1W_DWAI1 - Dword Untyped Atomic Integer Unary Write Only Operation MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Dword Untyped Atomic Integer Unary Operation

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 02h Project: All Format: Opcode Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 0h Project: All Format: Opcode Specifies that no return data is sent back to the thread.
	12	**SIMD Mode** Project: All Format: MDC_SM2R [CHV, BSW] Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Atomic Integer Operation** Project: All Format: MDC_AOP1 [CHV, BSW] Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS_SLM_A32 [CHV, BSW] Specifies the Binding Table Index for the message

End If

endif - End If

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

Description
The endif instruction terminates an if/else/endif block of code. It restores execution to the channels that were active prior to the if/else/endif block. The endif instruction is also used to hop out of nested conditionals by jumping to the end of the next outer conditional block when all channels are disabled.
The following table describes the 32-bit JIP. In GEN binary, JIP is at location src1 and must be of type D (signed DWord integer). JIP must be an immediate operand, it is a signed 32-bit number. This value is added to IP pre-increment.
Format: endif JIP

Restriction
Predication is not allowed.
The execution size must be the same for the if, else, and endif instructions of the same code block.

Syntax	Project
endif (exec_size) imm32	CHV, BSW

Pseudocode
Evaluate(WrEn); if (WrEn == 0) { // all channels false Jump(IP + JIP); }

Errata	Description
	An endif instruction must not be followed by any instruction requiring register indirect access on source operands.

Predication	Conditional Modifier	Saturation	Source Modifier
N	N	N	N

DWord	Bit	Description
0..3	127:96	**JIP** Project: CHV, BSW Format: S31 Jump Target Offset. The relative offset in bytes if a jump is taken for the instruction.
	95	**Source 0 Address Immediate [9] Sign Bit** Project: CHV, BSW

endif - End If		
	94:91	**Src1.SrcType** Project: CHV, BSW Format: SrcType [CHV, BSW]
	90:89	**Src1.RegFile** Project: CHV, BSW Format: RegFile [CHV, BSW]
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align16') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN16 [CHV, BSW]
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN1 [CHV, BSW]
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Extended Math Function

math - Extended Math Function	
Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The math instruction performs extended math function on the components in src0, or src0 and src1, and write the output to the channels of dst. The type of extended math function are based on the FC[3:0] encoding in the table below.

Format: [(pred)] math (exec_size) dst src0 src1 <FC>

Restriction
Accumulator access is allowed only for ieee macro functions.
The math instruction does not support indirect addressing modes.
The only supported rounding mode for math instruction is Round to Nearest Even.
INT DIV function does not support SIMD16. INT DIV function does not support simulataneous write to two registers. INT DIV function does not support source modifiers.
The FDIV function is not supported in ALT_MODE.
The math instruction can use GRF or immediates as source. The source formats for immediates must be one of the source formats supported by math operation.
DepCtrl must not be used.
The math instruction must use GRF as destination.
The supported regioning mode for math instruction is align1 and align16. The following restrictions apply for align1 mode: Scalar source is supported. Source and destination horizontal stride must be the same. Regioning must ensure Src.Vstride = Src.Width * Src.Hstride . Source and destination offset must be the same, except the case of scalar source.
Math Operation rules when float and half-floats are mixed between source or between source and destination operands. The half-float operand must be interleaved in the register for align1 and the source and destination register offset must be be the same to DW granularity. For align16, the half-float operand is allowed to be packed.
The execution size must be no more than 8 when half-floats are used in source or destination operand.
The source operand must not span two registers if the destination operand spans just one register Example: Case (a) // Allowed. The source must be strided by 2. the offset is allowed to select between lower/upper 16b math (8) r10:f r11.0<16;8,2>:hf 0x01 math (8) r10:f r11.1<16;8,2>:hf 0x01 math (8) r10:f r11.0<16;8,2>:hf r12.1<16;8,2>:hf 0x09 Case (b) // Allowed. The destination must be strided by 2. The offset is allowed to selecte between lower/upper 16b math (8) r10.0<2>:hf r11.0<8;8,1>:f math (8) r10.1<2>:hf r11.0<8;8,1>:f 0x01 math (8) r10.0<2>:hf r11.0<16;8,2>:hf r12.0<16;8,2>:hf 0x09 Case (c) // Allowed. Destination has stride of 2. The offset is allowed to select between uppoer/lower 16b math (8) r10.0<2>:hf r11.0<8;8,1>:f r12.1<16;8,2>:hf 0x09 math (8) r10.1<2>:hf r11.1<16;8,2>:hf r12.0<8;8,1>:f 0x09 Case (d) // Not Allowed. Destination is half-float but is not interleaved. math (8) r10.0<1>:hf r11.0<8;8,1>:f Case (e) // Not Allowed. Source is half-float but

math - Extended Math Function

not interleaved math (8) r10.0<2>:hf r11.0<8;8,1>:f r12.0<8;8,1>:hf 0x09 Case (f) // Not Allowed. Source operand spans 2 registers while destination spans one register. math (8) r83.8<1>:hf r12.4<4;4,1>:f null 0x02
Math Operation rules when half-floats are used on both source and destination operands. The execution size must be 8. The half-float source must be packed or interleaved. When interleaving, both source and destination must be interleaved. Example: Case (a) // Allowed. The source and destination are packed or interleaved math (8) r10.0:hf r11.0<8;8,1>:hf 0x01 math (8) r10.0<2>:hf r11.0<16;8,2>:hf 0x01 math (8) r10.8:hf r11.0<8;8,1>:hf 0x01 math (8) r10.8<2>:hf r11.0<16;8,2>:hf 0x01
For FDVI and POW, half-float destination data type is not allowed.
For one source math operations src1 must be NULL.

Syntax
[(pred)] math (exec_size) reg reg reg imm4

Pseudocode

```
Evaluate(WrEn);
for (n = 0; n < exec_size; n++) {
    if (WrEn.channel[n] == 1) {
        switch FC[3:0] {
            case 1h:
                dst.channel[n] = rcp(src0.channel[n]);
            case 2h:
                dst.channel[n] = log(src0.channel[n]);
            case 3h:
                dst.channel[n] = exp(src0.channel[n]);
            case 4h:
                dst.channel[n] = sqrt(src0.channel[n]);
            case 5h:
                dst.channel[n] = rsq(src0.channel[n]);
            case 6h:
                dst.channel[n] = sin(src0.channel[n]);
            case 7h:
                dst.channel[n] = cos(src0.channel[n]);
            case 9h: // src0 / src1
                dst.channel[n] = fdiv(src0.channel[n], src1.channel[n]);
            case Ah:
                dst.channel[n] = pow(src0.channel[n], src1/channel[n]);
            case Bh: // src0 / src1
                idiv(src0.channel[n], src1.channel[n]);
                dst.channel[n] = quotient;
                dst+1.channel[n] = remainder;
            case Ch:
                idiv(src0.channel[n], src1.channel[n]);
                dst.channel[n] = quotient;
            case Dh:
                idiv(src0.channel[n], src1.channel[n]);
                dst.channel[n] = remainder;
            }
        }
    }
}
```

math - Extended Math Function

Predication	Conditional Modifier	Saturation	Source Modifier [CHV, BSW]
Y	N	Y	Y

Src Types	Dst Types	Project
F	F	
D	D	
UD	UD	
F/HF	F/HF	CHV, BSW

DWord	Bit	Description
0..3	127:64	**RegSource** Format: EU_INSTRUCTION_SOURCES_REG_REG [CHV, BSW]
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:28	**Controls B** Format: EU_INSTRUCTION_CONTROLS_B [CHV, BSW]
	27:24	**Function Control (FC)** Format: FC [CHV, BSW]
	23:8	**Controls A** Format: EU_INSTRUCTION_CONTROLS_A [CHV, BSW]
	7	**Reserved** Format: MBZ
	6:0	**Opcode** Format: EU_OPCODE [CHV, BSW]

Find First Bit from LSB Side

fbl - Find First Bit from LSB Side

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The fbl instruction counts component-wise the number of LSB 0 bits before the first 1 bit in src0, storing that number in dst.

Format: [(pred)] fbl (exec_size) dst src0

Programming Notes
If src0 contains no 1 bits, store 0xFFFFFFFF in dst.

Restriction
No accumulator access, implicit or explicit.

Syntax
[(pred)] fbl (exec_size) reg reg [(pred)] fbl (exec_size) reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { UD cnt = 0; UD udScalar = src0.chan[n]; while ((udScalar & 1) == 0 && cnt != 32) { cnt ++; udScalar = udScalar » 1; } if (src0.chan[n] == 0x00000000) { dst.chan[n] = 0xFFFFFFFF; } else { dst.chan[n] = cnt; } } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

Src Types	Dst Types
UD	UD

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32 [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Find First Bit from MSB Side

fbh - Find First Bit from MSB Side

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

If src0 is unsigned, the fbh instruction counts component-wise the leading zeros from src0 and stores the resulting counts in dst. If src0 is signed and positive, the fbh instruction counts component-wise the leading zeros from src0 and stores the resulting counts in dst. If src0 is signed and negative, the fbh instruction counts component-wise the leading ones from src0 and stores the resulting counts in dst.

Format: [(pred)] fbh (exec_size) dst src0

Programming Notes
If src0 is zero, store 0xFFFFFFFF in dst.
If src0 is signed and is -1 (0xFFFFFFFF), store 0xFFFFFFFF in dst.

Restriction
No accumulator access, implicit or explicit.

Syntax
[(pred)] fbh (exec_size) reg reg [(pred)] fbh (exec_size) reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { UD cnt = 0; if (src0 is unsigned) { UD udScalar = src0.chan[n]; while ((udScalar & (1 « 31)) == 0 && cnt != 32) { cnt ++; udScalar = udScalar « 1; } if (src0.chan[n] == 0x00000000) { dst.chan[n] = 0xFFFFFFFF; } else { dst.chan[n] = cnt; } } else { // src0 is signed. D dScalar = src0.chan[n]; bit cval = dScalar[31]; while ((dScalar & (1 « 31)) == cval && cnt != 32) { cnt ++; dScalar = dScalar « 1; } if ((src0.chan[n] == 0xFFFFFFFF) || (src0.chan[n] == 0x00000000)) { dst.chan[n] = 0xFFFFFFFF; } else { dst.chan[n] = cnt; } } } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

Src Types	Dst Types
D, UD	UD

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32 [CHV, BSW]

fbh - Find First Bit from MSB Side		
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Fraction

frc - Fraction

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The frc instruction computes, component-wise, the truncate-to-minus-infinity fractional values of src0 and stores the results in dst. The results, in the range of [0.0, 1.0], are the fractional portion of the source data. The result is in the range [0.0, 1.0] irrespective of the rounding mode. Floating-point fraction computation follows the rules in the following tables, based on the current floating-point mode.

Format: [(pred)] frc[.cmod] (exec_size) dst src0

Workaround
When the Rounding Mode in cr0.0 is not Round Down, the result from frc must be followed by compare and select instructions to avoid a result of 1.0. Those latter instructions must use the :ud type. For example: cmp.ne.f0.0 null r4:ud 0x3f800000:ud (f0.0)sel r5:f r4:ud 0x3f7fffff:ud

Syntax
[(pred)] frc[.cmod] (exec_size) reg reg [(pred)] frc[.cmod] (exec_size) reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = src0.chan[n] - floor(src0.chan[n]); } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	N	Y

Src Types	Dst Types
F	F

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32 [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Goto

goto - Goto

Project:	CHV, BSW
Source:	Eulsa
Length Bias:	4

The goto instruction directs the instruction pointer to the offset specified by the UIP offset or to the next IP based on the BranchCtrl bit in the instruction. The active channels that are predicated on this instruction will take the IP + UIP path when BranchCtrl is set else the channels take IP + 1. The active channels that are not predicated on this instruction will be made inactive and waiting to be joined at the join IP. The join IP is IP + UIP when BranchCtrl is clear else it is the next IP.
When there are no active channels the instruction pointer will move to IP + JIP.
The goto instruction is used in conjunction with a join instruction. A goto deactivates some channels that are reactivated at some program-specified join instruction. See the join instruction for the activation rules.
The goto and join instructions enable unstructured program control flow. These instructions must be used with additional care where dangling channels can result without proper compiler checks, meaning that it is expected that programs will navigate through these paths to reactivate the channels. Hardware does not provide native checks or reconvergence.
The following table describes the two 32-bit instruction pointer offsets. Both the JIP and UIP are signed 32-bit numbers, added to IP pre-increment. In GEN binary, JIP and UIP are at locations src0 and src1 and must be of type DW (signed DWord integer).
If SPF is ON, none of the PcIP are updated.

Format: [(pred)] goto (exec_size) JIP UIP BranchCtrl

Restriction
Cannot have a goto in the body and the corresponding join in the function or the subroutine. Similarly the brd and brc.

Syntax
[(pred)] goto (exec_size) imm32 imm32 BranchCtrl

Pseudocode

Evaluate(WrEn);

```
for ( n = 0; n < exec_size; n++ ) { if ( WrEn.chan[n] ) { // for the predicated active channels if ( BranchCtrl ) { PcIP[n] = IP + UIP; } else { PcIP[n] = IP + 1; } } else { // join IP, for the active non predicated channels if ( BranchCtrl ) { PcIP[n] = IP + 1; } else { PcIP[n] = IP + UIP; } } } if ( BranchCtrl ) { // if (PcIP != (IP + UIP) ) { // for all channels if (PcIP != (IP + 1) ) { // for all channels Jump(IP + JIP); } else { Jump(IP + 1); } } else { Jump(IP + UIP); } } else { // if (PcIP != (IP + 1) ) { // for all channels Jump(IP + JIP); } else { Jump(IP + 1); } }
```

Errata	Description
	A goto instruction must not be followed by any instruction requiring register indirect access on source operands.

goto - Goto

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

DWord	Bit	Description
0..3	127:96	**JIP** Project: CHV, BSW Format: S31 The byte-aligned jump distance if a jump is taken for the channel.
	95:64	**UIP** Project: CHV, BSW Format: S31 The byte aligned jump distance if a jump is taken for the instruction.
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

GPGPU_CSR_BASE_ADDRESS

GPGPU_CSR_BASE_ADDRESS

Project:	CHV, BSW
Source:	PRM
Length Bias:	2

The GPGPU_CSR_BASE_ADDRESS command sets the base pointers for EU and L3 to Context Save and Restore EU State and SLM for **GPGPU** mid.

Programming Notes	Project
Execution of this command causes a full pipeline flush, thus its use should be minimized for higher performance. State and instruction caches are flushed on completion of the flush.	CHV, BSW
SW must always program PIPE_CONTROL with "CS Stall" and "Render Target Cache Flush Enable" set prior to programming GPGPU_CSR_BASE_ADDRESS command for GPGPU workloads i.e when pipeline select is GPGPU via PIPELINE_SELECT command. This is required to achieve better GPGPU preemption latencies for certain programming sequences. If programming PIPE_CONTROL has performance implications then preemption latencies can be trade off against performance by not implementing this programming note.	CHV, BSW

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: Opcode
	28:27	**Command SubType** Default Value: 0h GFXPIPE_COMMON Format: Opcode
	26:24	**3D Command Opcode** Default Value: 1h GFXPIPE_NONPIPELINED Format: Opcode
	23:16	**3D Command Sub Opcode** Default Value: 04h GPGPU_CSR_BASE_ADDRESS Format: Opcode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Total Length -2

Value	Name	Description	Project
1h	**[Default]**	Excludes DWord(0,1)	CHV, BSW

GPGPU_CSR_BASE_ADDRESS		
1..2 **Project:** CHV, BSW	63:12	**GPGPU CSR Base Address High** Project: CHV, BSW Format: GraphicsAddress[63:12] Specifies the 256K-byte aligned base address for GPGPU context GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].
	11:0	**Reserved** Project: All Format: MBZ

GPGPU_WALKER

GPGPU_WALKER

Project: CHV, BSW
Source: RenderCS
Length Bias: 2

Programming Notes
If the threads spawned by this command are required to observe memory writes performed by threads spawned from a previous command, software must precede this command with a command that performs a memory flush (e.g., MI_FLUSH).

DWord	Bit	Description
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: 1h GPGPU_WALKER Format: OpCode
	23:16	**SubOpcode** Default Value: 05h GPGPU_WALKER SubOp Format: OpCode
	15:11	**Reserved** Format: MBZ
	10	**Indirect Parameter Enable** Format: Enable If set, the values in DW 7, 10, 12 are ignored and replaced by the current values of the corresponding GPGPU_xxx MMIO registers: • GPGPU_DISPATCHDIMX (instead of DW7) • GPGPU_DISPATCHDIMY (instead of DW10) • GPGPU_DISPATCHDIMZ (instead of DW12)
	9	**Reserved** Format: MBZ

<table>
<tr><th colspan="3">GPGPU_WALKER</th></tr>
<tr><td></td><td>8</td><td>Predicate Enable
Format: | Enable
If set, this command is executed (or not) depending on the current value of the MI Predicate internal state bit. This command is ignored only if PredicateEnable is set and the Predicate state bit is 0.</td></tr>
<tr><td></td><td>7:0</td><td>DWord Length
Format: | =n Total Length - 2
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0Dh</td><td>DWORD_COUNT_n [Default]</td><td>Excludes DWord (0,1)</td></tr></table></td></tr>
<tr><td>1</td><td>31:8</td><td>Reserved</td></tr>
<tr><td></td><td>7:6</td><td>Reserved
Format: | MBZ</td></tr>
<tr><td></td><td>5:0</td><td>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.</td></tr>
<tr><td>2</td><td>31:17</td><td>Reserved
Format: | MBZ</td></tr>
<tr><td></td><td>16:0</td><td>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. If Indirect Data is enabled, it is assumed that CURBE is not being used except for cross-thread constant data. 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, the total size of Indirect data must be less than 63,488 (2048 URB lines - 64 lines for Interface Descriptors).
<table><tr><th>Workaround</th><th>Project</th></tr><tr><td>The indirect payload is limited to 4032 bytes or less.</td><td>CHV, BSW</td></tr></table></td></tr>
<tr><td>3</td><td>31:6</td><td>Indirect Data Start Address
Format: | IndirectObjectOffset[31:6]
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.
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>[0 - 512MB]</td><td></td><td>(Bits 31:29 MBZ)</td></tr></table></td></tr>
</table>

GPGPU_WALKER

DWord	Bit	Description
	5:0	**Reserved** Format: MBZ
4	31:30	**SIMD Size** This field determines the size of the payload and the number of bits of the execution mask that are expected. The kernel pointed to by the interface descriptor should match the SIMD declared here. <table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td>SIMD8</td><td>8 LSBs of the execution mask are used</td></tr><tr><td>1</td><td>SIMD16</td><td>16 LSBs used in execution mask</td></tr><tr><td>2</td><td>SIMD32</td><td>32 bits of execution mask used</td></tr></table>
	29:22	**Reserved** Format: MBZ
	21:16	**Thread Depth Counter Maximum** The maximum value of the thread depth counter. Since the counter starts at 0, the depth is this value + 1. **(Thread_Depth_Max+1)*(Thread_Height_Max+1)*(Thread_Width_Max+1)** must equal **Number of Threads in GPGPU Thread Group** in the Interface Descriptor.
	15:14	**Reserved** Format: MBZ
	13:8	**Thread Height Counter Maximum** Format: U6-1 The maximum value of the thread height counter. The height is this value + 1.
	7:6	**Reserved** Format: MBZ
	5:0	**Thread Width Counter Maximum** Format: U6-1 The maximum value of the thread width counter. The height is this value + 1.
5	31:0	**Thread Group ID Starting X** This is the initial value of the X component of the thread group. When X reaches the maximum value it rolls around to this value.
6	31:0	**Reserved** Format: MBZ
7	31:0	**Thread Group ID X Dimension** The X dimension of the thread group (maximum X is dimension -1)
8	31:0	**Thread Group ID Starting Y** This is the initial value of the Y component of the thread group. When Y reaches the maximum value it rolls around to this value.

GPGPU_WALKER		
9	31:0	**Reserved** Format: MBZ
10	31:0	**Thread Group ID Y Dimension** The Y dimension of the thread group (maximum Y is dimension -1)
11	31:0	**Thread Group ID Starting/Resume Z** This is the initial value of the Z component of the thread group.
12	31:0	**Thread Group ID Z Dimension** The Z dimension of the thread group (maximum Z is dimension -1)
13	31:0	**Right Execution Mask** The execution mask used for threads on the right (maximum X) of the thread group.
14	31:0	**Bottom Execution Mask** The execution mask used for threads on the bottom (maximum Y) of the thread group.

Half Precision HI8DS Render Target Write MSD

MSD_RTWH_HI8DS - Half Precision HI8DS Render Target Write MSD

Project:	CHV, BSW
Source:	Render Cache DataPort
Length Bias:	1
Family:	Other
Group:	Render Target R/W

DWord	Bit	Description
0	31	**Reserved**
	30	**Message Precision Subtype** Default Value: 1h Project: CHV, BSW Format: Opcode Half precision data message
	29	**Reserved** Project: All Format: MBZ Ignored
	28:25	**Message Length** Project: All Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Project: All Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the 2-register header.

MSD_RTWH_HI8DS - Half Precision HI8DS Render Target Write MSD

	18	**Reserved**
	17:14	**Message Type** Default Value: 0Ch Project: All Format: Opcode Render Target Write message
	13	**Reserved**
	12	**Last Render Target Select** Project: All Format: Enable This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero. **Programming Notes** When a pixel shader has render target writes at finer granularity than the dispatch rate, last render target write to a null surface must be present at the dispatch rate with this bit set. In particular, if a kernel is dispatched at pixel rate and it only writes to render targets at sample-rate, it must include a pixel-rate render target write to a null surface with Last Render Target Select bit enabled.
	11	**Slot Group Select** Project: All Format: MDC_RT_SGS [CHV, BSW] This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
	10:8	**Render Target Message Subtype** Default Value: 3h Project: All Format: Opcode SIMD8 dual source message. Use slots [15:8] for pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [31:24] are referenced instead of [15:8].
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Half Precision LO8DS Render Target Write MSD

MSD_RTWH_LO8DS - Half Precision LO8DS Render Target Write MSD

Project:	CHV, BSW
Source:	Render Cache DataPort
Length Bias:	1
Family:	Other
Group:	Render Target R/W

DWord	Bit	Description
0	31	**Reserved** Project: All Format: MBZ Ignored
	30	**Message Precision Subtype** Default Value: 1h Project: CHV, BSW Format: Opcode Half precision data message
	29	**Reserved**
	28:25	**Message Length** Project: All Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Project: All Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the 2-register header.

MSD_RTWH_LO8DS - Half Precision LO8DS Render Target Write MSD

	18	**Reserved**
	17:14	**Message Type** Default Value: 0Ch Project: All Format: Opcode Render Target Write message
	13	**Reserved**
	12	**Last Render Target Select** Project: All Format: Enable This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero. **Programming Notes** When a pixel shader has render target writes at finer granularity than the dispatch rate, last render target write to a null surface must be present at the dispatch rate with this bit set. In particular, if a kernel is dispatched at pixel rate and it only writes to render targets at sample-rate, it must include a pixel-rate render target write to a null surface with Last Render Target Select bit enabled.
	11	**Slot Group Select** Project: All Format: MDC_RT_SGS [CHV, BSW] This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
	10:8	**Render Target Message Subtype** Default Value: 2h Project: All Format: Opcode SIMD8 dual source message. Use slots [7:0] for pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [23:16] are referenced instead of [7:0].
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Half Precision REP16 Render Target Write MSD

MSD_RTWH_REP16 - Half Precision REP16 Render Target Write MSD

Project:	CHV, BSW
Source:	Render Cache DataPort
Length Bias:	1
Family:	Other
Group:	Render Target R/W

DWord	Bit	Description
0	31	**Reserved** Project: All Format: MBZ Ignored
	30	**Message Precision Subtype** Default Value: 1h Project: CHV, BSW Format: Opcode Half precision data message
	29	**Reserved** Project: All Format: MBZ Ignored
	28:25	**Message Length** Project: All Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Project: All Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.

MSD_RTWH_REP16 - Half Precision REP16 Render Target Write MSD

	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the 2-register header.
	18	**Reserved** Project: CHV, BSW Format: MBZ Ignored
	17:14	**Message Type** Default Value: 0Ch Project: All Format: Opcode Render Target Write message
	13	**Reserved** Project: CHV, BSW Format: MBZ Ignored
	12	**Last Render Target Select** Project: All Format: Enable This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero. **Programming Notes** When a pixel shader has render target writes at finer granularity than the dispatch rate, last render target write to a null surface must be present at the dispatch rate with this bit set. In particular, if a kernel is dispatched at pixel rate and it only writes to render targets at sample-rate, it must include a pixel-rate render target write to a null surface with Last Render Target Select bit enabled.
	11	**Slot Group Select** Project: All Format: MDC_RT_SGS [CHV, BSW] This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.

MSD_RTWH_REP16 - Half Precision REP16 Render Target Write MSD

	10:8	**Render Target Message Subtype** Default Value: 1h Project: All Format: Opcode SIMD16 Single source message with replicated data. Use slots [15:0] for pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [31:16] are referenced instead of [15:0].
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Half Precision SIMD8 Render Target Write MSD

MSD_RTWH_SIMD8 - Half Precision SIMD8 Render Target Write MSD

Project:	CHV, BSW
Source:	Render Cache DataPort
Length Bias:	1
Family:	Other
Group:	Render Target R/W

DWord	Bit	Description
0	31	**Reserved**
	30	**Message Precision Subtype** Default Value: 1h Project: CHV, BSW Format: Opcode Half precision data message
	29	**Reserved** Project: All Format: MBZ Ignored
	28:25	**Message Length** Project: All Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Project: All Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the 2-register header.

MSD_RTWH_SIMD8 - Half Precision SIMD8 Render Target Write MSD

	18	**Reserved**
	17:14	**Message Type** Default Value: 0Ch Project: All Format: Opcode Render Target Write message
	13	**Reserved**
	12	**Last Render Target Select** Project: All Format: Enable This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero. **Programming Notes** When a pixel shader has render target writes at finer granularity than the dispatch rate, last render target write to a null surface must be present at the dispatch rate with this bit set. In particular, if a kernel is dispatched at pixel rate and it only writes to render targets at sample-rate, it must include a pixel-rate render target write to a null surface with Last Render Target Select bit enabled.
	11	**Slot Group Select** Project: All Format: MDC_RT_SGS [CHV, BSW] This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
	10:8	**Render Target Message Subtype** Default Value: 4h Project: All Format: Opcode SIMD8 single source message. Use slots [7:0] for pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [23:16] are referenced instead of [7:0].
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Half Precision SIMD16 Render Target Write MSD

MSD_RTWH_SIMD16 - Half Precision SIMD16 Render Target Write MSD

Project:	CHV, BSW
Source:	Render Cache DataPort
Length Bias:	1
Family:	Other
Group:	Render Target R/W

DWord	Bit	Description
0	31	**Reserved** Project: All Format: MBZ Ignored
	30	**Message Precision Subtype** Default Value: 1h Project: CHV, BSW Format: Opcode Half precision data message
	29	**Reserved** Project: All Format: MBZ Ignored
	28:25	**Message Length** Project: All Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Project: All Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.

MSD_RTWH_SIMD16 - Half Precision SIMD16 Render Target Write MSD

	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the 2-register header.
	18	**Reserved** Project: CHV, BSW Format: MBZ Ignored
	17:14	**Message Type** Default Value: 0Ch Project: All Format: Opcode Render Target Write message
	13	**Reserved** Project: CHV, BSW Format: MBZ Ignored
	12	**Last Render Target Select** Project: All Format: Enable This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero. **Programming Notes** When a pixel shader has render target writes at finer granularity than the dispatch rate, last render target write to a null surface must be present at the dispatch rate with this bit set. In particular, if a kernel is dispatched at pixel rate and it only writes to render targets at sample-rate, it must include a pixel-rate render target write to a null surface with Last Render Target Select bit enabled.
	11	**Slot Group Select** Project: All Format: MDC_RT_SGS [CHV, BSW] This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.

MSD_RTWH_SIMD16 - Half Precision SIMD16 Render Target Write MSD

	10:8	**Render Target Message Subtype** Default Value: 0h Project: All Format: Opcode SIMD16 Single source message. Use slots [15:0] for pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [31:16] are referenced instead of [15:0].
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Halt

halt - Halt

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

Description
The halt instruction temporarily suspends execution for all enabled compute channels. Upon execution, the enabled channels are sent to the instruction at (IP + UIP), if all channels are enabled at HALT, jump to the instruction at (IP + JIP). If the halt instruction is not inside any conditional code block, the values of JIP and UIP should be the same. If the halt instruction is inside a conditional code block, the UIP should be the end of the program and the JIP should be the end of the inner most conditional code block. The UIP must point to a HALT Instruction. If SPF is ON, the UIP must be used to update IP; JIP is not used in this case.
The following table describes the two 32-bit instruction pointer offsets. Both the JIP and UIP are signed 32-bit numbers, added to IP pre-increment. In GEN binary, JIP and UIP are at locations src0 and src1 and must be of type DW (signed DWord integer). When the offsets are immediate, src0 regfile must be immediate.
Format: [(pred)] halt (exec_size) JIP UIP

Restriction
dst and src0 must be NULL.

Syntax	Project
[(pred)] halt (exec_size) imm32 imm32	CHV, BSW

Pseudocode
Evaluate(WrEn); for (n = 0; n < 32; n++) { if (WrEn.channel[n]) { PcIP[n] = IP + UIP; else { PcIP[n] = IP + 1; } } if (PcIP != (IP + 1)) { // for all channels Jump(IP + JIP); }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

DWord	Bit	Description
0..3	127:96	**JIP** Project: CHV, BSW Format: S31 The byte-aligned jump distance if a jump is taken for the channel.
	95:64	**UIP** Project: CHV, BSW Format: S31 The byte aligned jump distance if a jump is taken for the instruction.

halt - Halt		
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

HCP_BSD_OBJECT

HCP_BSD_OBJECT

Project:	CHV, BSW
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: U12 (Excludes Dwords 0, 1). Value: 1h, Name: (blank)

<table>
<tr><th colspan="3">HCP_BSD_OBJECT</th></tr>
<tr><td>1</td><td>31:0</td><td>Indirect BSD Data Length
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>U32</td></tr></table>
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.
<table><tr><th>Value</th><th>Name</th><th>Description</th><th>Project</th></tr>
<tr><td>[0,268435455]</td><td>Data_Length_with_28_bits_only</td><td>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.</td><td>CHV, BSW</td></tr></table></td></tr>
<tr><td rowspan="2">2</td><td>31:29</td><td>Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>28:0</td><td>Indirect Data Start Address
<table><tr><td>Format:</td><td>U29</td></tr></table>
Specifies the byte-aligned graphics memory starting address of the slice bit stream relative to the BSD Indirect Object Base Address.</td></tr>
</table>

HCP_IND_OBJ_BASE_ADDR_STATE

HCP_IND_OBJ_BASE_ADDR_STATE

Project:	CHV, BSW
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_IND_OBJ_BASE_ADDR_STATE command is used to define the indirect object base address of the stream in graphics memory. This is a frame level command. (Is it frame or picture level?)
This is a picture level state command and is issued in both encoding and decoding processes.

Compressed Header Format

Fields	Bits	
Bin	0	HuC/Kernel Binarized Syntax
Probability select	1	0 ->indicates probability 128 1 ->indicates probability 256
	Repeat to pack a Cacheline	

Partition1 and TileSize record

Fields	Bits	
Tile Size	31:0	Partition1 Size is 16-bit value, Tile Size is 32-bit value
AddressOffset	63:32	Cacheline Address Offset to be Modified
Offset	69:64	Byte offset to be Modified
16-bit vs 32-bit update	70	0: Update 16-bit; 1: Update 32-bit
Reserved	511:71	

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

HCP_IND_OBJ_BASE_ADDR_STATE

DWord	Bit	Description
	22:16	**Media Instruction Command** Default Value: 3h HCP_IND_OBJ_BASE_ADDR_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 4h; Name: ; Project: CHV, BSW
1..2 **Project:** All	63:0	**HCP Indirect Bitstream Object Base Address** Project: All Format: SplitBaseAddress4KByteAligned Specifies the 4K-byte aligned memory base address for the read-only indirect data object pointed in the HCP_BSD_OBJECT command for fetching (reading) the compressed Slice Data.
3 **Project:** All	31:0	**HCP Indirect Bitstream Object Memory Address Attributes** Project: All Format: MemoryAddressAttributes [CHV, BSW]
4..5 **Project:** All	63:0	**HCP Indirect Bitstream Object Access Upper Bound** Project: All Format: SplitBaseAddress4KByteAligned Decoder only. This field specifies the 4K-byte aligned maximum memory address access by the indirect data object in the HCP_BSD_OBJECT command for the slice bit stream. Indirect data accessed at this address or greater will cause the HCP to stop issuing requests to the GAC and the BSP VLD will then only receive zeros until a slice done is received. Setting this field to 0 will cause this range to be ignored by the HCP.
6..13 **Project:** CHV, BSW	31:0	**Reserved** Project: CHV, BSW Format: MBZ

HCP_PIC_STATE

HCP_PIC_STATE

Project:	CHV, BSW
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: (blank)
1	31:26	**Reserved** Project: CHV, BSW Format: MBZ

HCP_PIC_STATE

DWord	Bit	Description
	25:16	**FrameHeightInMinCbMinus1**

Project:	CHV, BSW
Format:	U10

Specifies the height of each decoded picture in units of minimum coding block size.

Value	Name
[0-4122]	

Programming Notes	Project
The decoded picture height in units of luma samples equals • (FrameHeightInMinCbMinus1 + 1) * • (1 « (log2_min_coding_block_size_minus3 + 3))	
In CHV, BSW, the maximum frame height being support is 7968 pixels only	CHV, BSW

DWord	Bit	Description
	14:10	**Reserved**
	9:0	**FrameWidthInMinCbMinus1**

Project:	CHV, BSW
Format:	U10

Specifies the width of each decoded picture in units of minimum coding block size.

Value	Name
[0-4122]	

Programming Notes
The decoded picture width in units of luma samples equals • (FrameWidthInMinCbMinus1 + 1) * • (1 « (log2_min_coding_block_size_minus3 + 3))

DWord	Bit	Description
2	31:12	**Reserved**

Project:	
Format:	MBZ

DWord	Bit	Description
	11:10	**MaxPCMSize**

Format:	U2

Specifies the largest allowed PCM coding block size.

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

Programming Notes
LCU is restricted based on the picture size.

HCP_PIC_STATE

DWord	Bit	Description
	1:0	**MinCUSize**

Format:	U2

Specifies the smallest coding block size.

Value	Name
3	64x64
2	32x32
1	16x16
0	8x8

DWord	Bit	Description
3	31:3	**Reserved**

Project:	
Format:	MBZ

DWord	Bit	Description
	2	**InsertTestFlag**

Format:	U1

Value	Name
0h	**[Default]**
1h	

Programming Notes
Encoder only. 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 } / 3Modified 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

DWord	Bit	Description
	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	Name
0	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.

DWord	Bit	Description
	0	**ColPicIsI**

HCP_PIC_STATE

<table>
<tr><td></td><td></td><td>
<table><tr><td>Format:</td><td>U1</td></tr></table>
Specifies that the collocated picture is comprised solely of I slices and that there are no P or B slices in the picture.
<table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>Collocated picture has at least one P or B slice</td></tr></table>
<table><tr><th>Programming Notes</th></tr><tr><td>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.</td></tr></table>
</td></tr>
<tr><td>4</td><td>31:28</td><td>Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>27</td><td>CU packet structure</td></tr>
<tr><td></td><td>26</td><td>strong_intra_smoothing_enable_flag
<table><tr><td>Format:</td><td>U1</td></tr></table></td></tr>
<tr><td></td><td>25</td><td>transquant_bypass_enable_flag
<table><tr><td>Format:</td><td>Enable</td></tr></table>
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td>Disable</td><td>cu_transquant_bypass is not supported</td></tr><tr><td>1</td><td>Enable</td><td>cu_transquant_bypass is supported</td></tr></table></td></tr>
<tr><td></td><td>24</td><td>Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>23</td><td>amp_enabled_flag
<table><tr><td>Format:</td><td>Enable</td></tr></table>
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td>Disable</td><td>Asymmetric motion partitions cannot be used in coding tree blocks.</td></tr><tr><td>1</td><td>Enable</td><td>Support asymmetric motion partitions, i.e. PartMode equal to PART_2NxnU, PART_2NxnD, PART_nLx2N, or PART_nRx2N.</td></tr></table></td></tr>
<tr><td></td><td>22</td><td>transform_skip_enabled_flag
<table><tr><td>Format:</td><td>Enable</td></tr></table>
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td>Disable</td><td>transform_skip_flag is not supported in the residual coding</td></tr><tr><td>1</td><td>Enable</td><td>transform_skip_flag is supported</td></tr></table></td></tr>
<tr><td></td><td>21</td><td>BottomField</td></tr>
</table>

HCP_PIC_STATE

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**

Format:	U1

16 **entropy_coding_sync_enabled_flag**

Format:	U1

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

HCP_PIC_STATE

		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 **Programming Notes** For encoder, always set to 0 to disable this (Intel restriction).
	8	**pcm_loop_filter_disable_flag** Format: U1
	7:6	**diff_cu_qp_delta_depth (or named as max_dqp_depth)** Format: U2 **Programming Notes** cu_qp_delta_enabled_flag = 1 and Max_DQP_Level = 0 is supported. 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 LCU, no change across CU. Must be zero for encoder
	5	**cu_qp_delta_enabled_flag** Format: U1 **Value** | **Name** | **Description** 0 | Disable | Does not allow QP change at CU level, the same QP is used for the entire LCU. 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. **Programming Notes** cu_qp_delta_enabled_flag = 1 and Max_DQP_Level = 0 is supported.
	4	**pcm_enabled_flag** Format: U1

HCP_PIC_STATE		
	3	**sample_adaptive_offset_enabled_flag** Format: U1 **Programming Notes** Restriction: HW does not support SAO filtering for LCU size 16x16 Should be set to 0 if slice_sao_luma_flag==0 and slice_sao_chroma_flag==0.
	2:0	**Reserved** Format: MBZ
5	31:30	**Reserved** Format: MBZ
	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: U5
	4:0	**pic_cb_qp_offset** Format: U5
6..18	31:0	**Reserved** Project: CHV, BSW Format: MBZ

HCP_PIPE_BUF_ADDR_STATE

HCP_PIPE_BUF_ADDR_STATE

Project:	CHV, BSW
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 state command provides the memory base addresses for the row store buffer and reconstructed picture output buffers required by the HCP.
This is a picture level state command and is shared by both encoding and decoding processes.

Programming Notes
All pixel surface addresses must be 4K byte aligned. There is a max of 8 Reference Picture Buffer Addresses, and all share the same third address DW in specifying 48-bit address.

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: 2h HCP_PIPE_BUF_ADDR_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 5Ch; Name: (blank)

HCP_PIPE_BUF_ADDR_STATE		
1..2	63:0	**Decoded Picture** Format: SplitBaseAddress4KByteAligned Frame buffer address for the final decoded picture YUV output.
3	31:0	**Decoded Picture Memory Address Attributes** Project: CHV, BSW Format: MemoryAddressAttributes [CHV, BSW]
4..5	63:0	**Deblocking Filter Line Buffer** Format: SplitBaseAddress64ByteAligned Base address of the filter line buffer (read/write) used by the Deblocking Filter.
6	31:0	**Deblocking Filter Line Buffer Memory Address Attributes** Project: CHV, BSW Format: MemoryAddressAttributes [CHV, BSW]
7..8	63:0	**Deblocking Filter Tile Line Buffer** Format: SplitBaseAddress64ByteAligned Base address of the tile line buffer (read/write) used by the Deblocking Filter.
9	31:0	**Deblocking Filter Tile Line Buffer Memory Address Attributes** Project: CHV, BSW Format: MemoryAddressAttributes [CHV, BSW]
10..11	63:0	**Deblocking Filter Tile Column Buffer** Format: SplitBaseAddress64ByteAligned Base address of the tile column buffer (read/write) used by the Deblocking Filter.
12	31:0	**Deblocking Filter Tile Column Buffer Memory Address Attributes** Project: CHV, BSW Format: MemoryAddressAttributes [CHV, BSW]
13..14	63:0	**Metadata Line Buffer** Format: SplitBaseAddress64ByteAligned Base address for the Metadata Line buffer.
15	31:0	**Metadata Line Buffer Memory Address Attributes** Project: CHV, BSW Format: MemoryAddressAttributes [CHV, BSW]
16..17	63:0	**Metadata Tile Line Buffer** Format: SplitBaseAddress64ByteAligned Base address for the Metadata Tile Line buffer.

HCP_PIPE_BUF_ADDR_STATE		
18	31:0	**Metadata Tile Line Buffer Memory Address Attributes** Project: CHV, BSW Format: MemoryAddressAttributes [CHV, BSW]
19..20	63:0	**Metadata Tile Column Buffer** Format: SplitBaseAddress64ByteAligned Base address for the Metadata Tile Column buffer.
21	31:0	**Metadata Tile Column Buffer Memory Address Attributes** Project: CHV, BSW Format: MemoryAddressAttributes [CHV, BSW]
22..23	63:0	**SAO Line Buffer** Format: SplitBaseAddress64ByteAligned Base address for the SAO Line buffer.
24	31:0	**SAO Line Buffer Memory Address Attributes** Project: CHV, BSW Format: MemoryAddressAttributes [CHV, BSW]
25..26	63:0	**SAO Tile Line Buffer** Format: SplitBaseAddress64ByteAligned Base address for the SAO Tile Line buffer.
27	31:0	**SAO Tile Line Buffer Memory Address Attributes** Project: CHV, BSW Format: MemoryAddressAttributes [CHV, BSW]
28..29	63:0	**SAO Tile Column Buffer** Format: SplitBaseAddress64ByteAligned Base address for the SAO Tile Column buffer.
30	31:0	**SAO Tile Column Buffer Memory Address Attributes** Project: CHV, BSW Format: MemoryAddressAttributes [CHV, BSW]
31..32	63:0	**Current Motion Vector Temporal Buffer** Format: SplitBaseAddress64ByteAligned Base address for the Current Motion Vector Temporal buffer.
33	31:0	**Current Motion Vector Temporal Buffer Memory Address Attributes** Project: CHV, BSW Format: MemoryAddressAttributes [CHV, BSW]

HCP_PIPE_BUF_ADDR_STATE

34..35	63:0	**Reserved** Format: MBZ
36	31:0	**Reserved** Project: CHV, BSW Format: MBZ
37..52	63:0	**Reference Picture Base Address (RefAddr[0-7])** Format: SplitBaseAddress64ByteAligned[8] Base address of the reference picture buffer. **Programming Notes** Must be 4k byte aligned.
53	31:0	**Reference Picture Base Address Memory Address Attributes** Project: CHV, BSW Format: MemoryAddressAttributes [CHV, BSW]
54..59	31:0	**Reserved**
60..61	63:0	**Reserved** Project: CHV, BSW
62	31:0	**Reserved**
63..65 **Project:** CHV, BSW	31:0	**Reserved** Project: CHV, BSW Format: MBZ
66..81	63:0	**Collocated Motion Vector Temporal Buffer[0-7]** Format: SplitBaseAddress64ByteAligned[8] Base address for the Collocated Motion Vector Temporal buffer. Specifies the 64 byte aligned buffer address for Collocated Motion Vector Temporal Buffer. For VP9, only index 0 is required to have a valid address.
82	31:0	**Collocated Motion Vector Temporal Buffer[0-7] Memory Address Attributes** Project: CHV, BSW Format: MemoryAddressAttributes [CHV, BSW]
83..84	63:0	**VP9 Probability Buffer Read/Write** Format: SplitBaseAddress64ByteAligned Specifies the 64 byte aligned buffer address for VP9 Probability Buffer. Hardware reads in the probability for decode and write out the modified probability for future frames. Driver needs to program the Initial VP9 Probability for decoding the current frame. For Key Frame, it should contain the default Key Frame Probability. For non-Key Frame, it could be a default (non-Key) or one of the 8 Reference Buffers Probability. Driver must provide a valid Initial VP9 Probability buffer.

HCP_PIPE_BUF_ADDR_STATE		
85	31:0	**VP9 Probability Buffer Read/Write Memory Address Attributes** Format: MemoryAddressAttributes [CHV, BSW]
86..87	63:0	**VP9 Segment ID Buffer Read/Write** Format: SplitBaseAddress64ByteAligned Specifies the 64 byte aligned buffer address for VP9 SegmentID buffer. This should contain the writeout SegmentID from previous frame and will be used to predict SegmentID for the current frame. Hardware will write out SegmentID of the current frame in the same address for the next frame.
88	31:0	**VP9 Segment ID buffer Read/Write Memory Address Attributes** Format: MemoryAddressAttributes [CHV, BSW]
89..90	63:0	**VP9 HVD Line Rowstore Buffer Read/Write** Format: SplitBaseAddress64ByteAligned Specifies the 64 byte aligned buffer address for HVD Line Rowstore Buffer (bitstream decoder).
91	31:0	**VP9 HVD Line Rowstore buffer Read/Write Memory Address Attributes** Format: MemoryAddressAttributes [CHV, BSW]
92..93	63:0	**VP9 HVD Tile Rowstore Buffer Read/Write** Format: SplitBaseAddress64ByteAligned Specifies the 64 byte aligned buffer address for HVD Tile Rowstore Buffer (bitstream decoder).
94	31:0	**VP9 HVD Tile Rowstore buffer Read/Write Memory Address Attributes** Format: MemoryAddressAttributes [CHV, BSW]
95..96	63:48	**MBZ: Reserved**
97	31:0	**SAO Streamout Data Destination buffer Read/Write Memory Address Attributes** Format: MemoryAddressAttributes [CHV, BSW]

HCP_PIPE_MODE_SELECT

HCP_PIPE_MODE_SELECT

Project:	CHV, BSW
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: 2h; Name: Value_2; Project: CHV, BSW
1	31:24	**Reserved**

HCP_PIPE_MODE_SELECT

	23:17	**Reserved** Format: MBZ
	16:13	**Reserved** Project: CHV, BSW Format: MBZ
	12	**Reserved** Project: CHV, BSW
	11	**Reserved** Project: CHV, BSW Format: MBZ
	10	**Reserved** Project: CHV, BSW
	9	**Reserved** Project: CHV, BSW
	8	**Reserved** Project: CHV, BSW
	7:5	**Codec Standard Select**
	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
	2	**Reserved** Project: CHV, BSW
	1:0	**Reserved** Project: CHV, BSW Format: MBZ

Codec Standard Select (bits 7:5):

Value	Name
0	HEVC
1	VP9

Pic Status/Error Report Enable (bit 3):

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.

HCP_PIPE_MODE_SELECT		
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.
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.

DWord 2 values:

Value	Name	Project
0	Disable	CHV, BSW

DWord 3 values:

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_QM_STATE

HCP_QM_STATE

Project:	CHV, BSW
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_QM_STATE command loads the custom HEVC quantization tables into local RAM and may be issued up to 20 times: 3x Colour Component plus 2x intra/inter plus 4x SizeID minus 4 for the 32x32 chroma components.
When the scaling_list_enable_flag is set to disable, the scaling matrix is still sent to the decoder, and with all entries programmed to the same value = 16.
This is a picture level state command and is issued in both encoding and decoding processes.

Dwords 2-17 form a table for the DCT coefficients, 4 8-bit coefficients/DWord.

- Size 4x4 for SizeID0, DWords 2-5.
- Size 8x8 for SizeID1/2/3, DWords 2-17.

SizeID 0 (Table 4-10)

4x4	[31:24]	[23:16]	[15:8]	[7:0]
DWord 2	AC(0,3)	AC(0,2)	AC(0,1)	DC
DWord 3	AC(1,3)	AC(1,2)	AC(1,1)	AC(1,0)
DWord 4	AC(2,3)	AC(2,2)	AC(2,1)	AC(2,0)
DWord 5	AC(3,3)	AC(3,2)	AC(3,1)	AC(3,0)

SizeID 1, 2, 3 (Table 4-11)

8x8	[31:24]	[23:16]	[15:8]	[7:0]	[31:24]	[23:16]	[15:8]	[7:0]
DWord 3,2	AC(0,7)	AC(0,6)	AC(0,5)	AC(0,4)	AC(0,3)	AC(0,2)	AC(0,1)	DC
DWord 5,4	AC(1,7)	AC(1,6)	AC(1,5)	AC(1,4)	AC(1,3)	AC(1,2)	AC(1,1)	AC(1,0)
DWord 7,6	AC(2,7)	AC(2,6)	AC(2,5)	AC(2,4)	AC(2,3)	AC(2,2)	AC(2,1)	AC(2,0)
DWord 9,8	AC(3,7)	AC(3,6)	AC(3,5)	AC(3,4)	AC(3,3)	AC(3,2)	AC(3,1)	AC(3,0)
DWord 11,10	AC(4,7)	AC(4,6)	AC(4,5)	AC(4,4)	AC(4,3)	AC(4,2)	AC(4,1)	AC(4,0)
DWord 13,12	AC(5,7)	AC(5,6)	AC(5,5)	AC(5,4)	AC(5,3)	AC(5,2)	AC(5,1)	AC(5,0)
DWord 15,14	AC(6,7)	AC(6,6)	AC(6,5)	AC(6,4)	AC(6,3)	AC(6,2)	AC(6,1)	AC(6,0)
DWord 17,16	AC(7,7)	AC(7,6)	AC(7,5)	AC(7,4)	AC(7,3)	AC(7,2)	AC(7,1)	AC(7,0)

DWord	Bit	Description	
0	31:29	**Command Type**	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode

HCP_QM_STATE

	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: 4h HCP_QM_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 10h — Name: (none)
1	31	**Chicken Bit Transquant Bypass Clamp Disable** Format: MBZ
	30:13	**Reserved** Format: MBZ
	12:5	**DC Coefficient** Format: U8 Specifies the 8-bit DC coefficient for SizeID 2 and 3. **Programming Notes** The DC Coefficient must be set to zero for SizeID 0 and 1. The DC Coefficient must be set to scaling_list_dc_coef_minus8 + 8 for SizeID 2 and 3.
	4:3	**Color Component** Format: U2 Encoder: When RDOQ is enabled, scaling list for all 3 color components must be same. So this field is set to always 0. Value 0: Luma Value 1: Chroma Cb Value 2: Chroma Cr Value 3: Reserved

HCP_QM_STATE		
	2:1	**SizeID** Format: U2 Value | Name | Description 0 | 4x4 | 1 | 8x8 | 2 | 16x16 | 3 | 32x32 | (Illegal Value for Colour Component Chroma Cr and Cb.)
	0	**Prediction Type** Format: U1 Value | Name 0 | Intra 1 | Inter
2..17	511:0	**See Tables 4-10 and 4-11**

HCP_REF_IDX_STATE

HCP_REF_IDX_STATE

Project:	CHV, BSW
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.

Unlike AVC, HEVC allows 16 reference idx entries in each of the L0 and L1 list for a progressive picture. Hence, a max total 32 reference idx in both lists together. The same when the picture is a field picture. Regardless the number of reference idx entries, there are only max 8 reference pictures exist at any one time. Multiple reference idx can point to the same reference picture and can optionally pic a top or bottom field, or frame.

For P-Slice, this command is issued only once, representing L0 list. For B-Slice, this command can be issued up to two times, one for L0 list and one for L1 list.

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: 12h HCP_REF_IDX_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 10h — Name: (blank)

HCP_REF_IDX_STATE

DWord	Bit	Description
1	31:5	**Reserved**
	4:1	**num_ref_idx_l[RefPicListNum]_active_minus1** Format: U4 num_ref_idx_l[*RefPicListNum*]_active_minus1 **Value** / **Name** [0-14] /
	0	**RefPicListNum** Format: U1 **Value** / **Name** 0 / Reference Picture List 0 1 / Reference Picture List 1
2..17	31:16	**Reserved** Format: MBZ
	15	**list_entry_lX[i]: bottom_field_flag** Format: U1 Where X is the RefPicListNum and i is the list entry number 0 through 15. DW2 corresponds to i=0, DW17 corresponds to i=15. **Value** / **Name** 0 / Bottom Field 1 / Top Field **Programming Notes** Not supported in encoder mode.
	14	**list_entry_lX[i]: field_pic_flag** Format: U1 Where X is the RefPicListNum and i is the list entry number 0 through 15. DW2 corresponds to i=0, DW17 corresponds to i=15. **Value** / **Name** 0 / Video Frame 1 / Video Field **Programming Notes** Not supported in encoder mode.
	13	**list_entry_lX[i]: LongTermReference** Format: U1 Where X is the RefPicListNum and i is the list entry number 0 through 15. DW2 corresponds to i=0, DW17 corresponds to i=15.

HCP_REF_IDX_STATE

		Value: 0 = Short term reference; 1 = Long term reference
	12	**luma_weight_lX_flag[i]** Format: U1 Where X is the RefPicListNum and i is the list entry number 0 through 15. DW2 corresponds to i=0, DW17 corresponds to i=15. Value 0: Default weighted prediction for luma Value 1: Explicit weighted prediction for Luma
	11	**chroma_weight_lX_flag[i]** Format: U1 Where X is the RefPicListNum and i is the list entry number 0 through 15. DW2 corresponds to i=0, DW17 corresponds to i=15. Value 0: Default weighted prediction for Chroma Value 1: Explicit weighted prediction for Chroma
	10:8	**list_entry_lX[i]: Reference Picture Frame ID (RefAddr[0-7])** Format: U3 Where X is the RefPicListNum and i is the list entry number 0 through 15. DW2 corresponds to i=0, DW17 corresponds to i=15. The reference picture frame ID identifies the reference picture associated with the base address defined in **Reference Picture Address (RefAddr[0-7])** in the HCP_PIPE_BUF_ADDR_STATE command.
	7:0	**list_entry_lX[i]: Reference Picture tb Value** Format: U8 Where X is the RefPicListNum and i is the list entry number 0 through 15. DW2 corresponds to i=0, DW17 corresponds to i=15. clip(-128,127, CurrentPOC - RefPOC), where RefPOC is the POC value of the reference picture. 8-bit signed. See the "Derivation process for temporal luma motion vector prediction" in the HEVC standard.

HCP_SLICE_STATE

HCP_SLICE_STATE

Project: CHV, BSW

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: 4h; Name: ; Project: CHV, BSW
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.

HCP_SLICE_STATE

	15:9	**Reserved** Format: MBZ
	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** Project: CHV, BSW
	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	31:24	**Reserved** Format: MBZ
	23	**Reserved** Project: Format: MBZ
	22	**Reserved** Project: CHV, BSW
	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.

HCP_SLICE_STATE

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

HCP_SLICE_STATE

15h	-11
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
10h	10
11h	11
12h	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

HCP_SLICE_STATE

DWord	Bit	Description
	4	**dependent_slice_flag** Format: U1 Decoder only.
	2	**LastSliceofPic** Format: U1 Value 0: Not the last slice of the picture Value 1: Last slice of the picture
	1:0	**slice_type** Format: U2 Value 0: B-slice Value 1: P-slice Value 2: I-slice Value 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 0: 0 Value 1: 1 Value 2: 2 Value 3: 3 Value 4: 4 **Programming Notes** The valid value is from 0 to 4 (MaxNumMergeCand = 5 - five_minus_max_num_merge_cand -1)

HCP_SLICE_STATE

	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 **Programming Notes** Note: For encoder, all Slices must have same setting within a picture
	11	**slice_sao_chroma_flag** Format: U1 **Programming Notes** Note: For encoder, all Slices must have same setting within a picture
	10	**slice_loop_filter_across_slices_enabled_flag** Format: U1 **Programming Notes** Note: For the encoder, this bit must be set to zero (hardware restriction).
	9	**Reserved**
	8:5	**slice_beta_offset_div2 or (final Beta_Offset_div2 Encoder)** Format: S3 Deblocking filter beta offset. Specified in 2's comp. **Value** / **Name** [1101b,0011b] / [-3,3]

HCP_SLICE_STATE		
	4:1	**slice_tc_offset_div2 or (final tc_offset_div2 Encoder)** Format: S3 Deblocking filter tc offset. Specified in 2's comp. Value: [1101b,0011b] — Name: [-3,3]
	0	**slice_header_disable_deblocking_filter_flag** Format: U1
5	31:16	**Reserved** Format: MBZ
	15:0	**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).
6..8	31:0	**Reserved** Project: CHV, BSW Format: MBZ

HCP_SURFACE_STATE

HCP_SURFACE_STATE

Project:	CHV, BSW
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: U12 (Excludes Dwords 0, 1). Value: 1h — Name:
1	31:28	**Surface Id** Format: U4

<table>
<tr><th colspan="3">HCP_SURFACE_STATE</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>HEVC: For current decoded Picture</td><td>8-bit uncompressed data</td></tr>
<tr><td>1h</td><td>Source Input Picture (encoder)</td><td>8-bit uncompressed data</td></tr>
<tr><td>2h</td><td>Prev Reference Picture</td><td>(VP9 only) Previous Reference</td></tr>
<tr><td>3h</td><td>Golden Reference Picture</td><td>(VP9 only) Golden Reference</td></tr>
<tr><td>4h</td><td>AltRef Reference Picture</td><td>(VP9 only) AltRef Reference</td></tr>
<tr><td>5h</td><td>Reference Picture</td><td></td></tr>
</table>
</td></tr>
<tr><td></td><td>27:17</td><td>Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table>
</td></tr>
<tr><td></td><td>16:0</td><td>Surface Pitch Minus1
<table><tr><td>Format:</td><td>U17-1</td></tr></table>
This field specifies the surface pitch in (#Bytes - 1).
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>

For tiled surfaces, the pitch must be a multiple of the tile width.
If Half Pitch for Chroma is set, this field must be a multiple of two tile widths for tiled surfaces, or a multiple of 2 bytes for linear surfaces.

For Y-tiled surfaces: Range = [127, 524287]->[128B,256KB] = [1 tile, 2048 tiles].
</td></tr>
</table>
</td></tr>
<tr><td>2</td><td>27:15</td><td>Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table>
</td></tr>
<tr><td></td><td>14:0</td><td>Y Offset for U(Cb) in pixel
<table><tr><td>Format:</td><td>U15_Pixel_Row_Offset</td></tr></table>
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.
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>

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) - 32 pixel aligned
TileYF (New 4k) - 64 pixel aligned
TileYS (64k) - 256 pixel aligned

</td></tr>
</table>
</td></tr>
</table>

HCP_TILE_STATE

HCP_TILE_STATE

Project:	CHV, BSW
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.

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: 11h HCP_TILE_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: U12 (Excludes Dwords 0, 1). Value: Bh — Name:
1	31:10	**Reserved** Format: MBZ
	9:5	**NumTileColumnsMinus1** Format: U5 Specifies the number of tile columns in Ctbs per picture.

HCP_TILE_STATE		
	4:0	**NumTileRowsMinus1** Format: U5 Specifies the number of tile rows in Ctbs per picture.
2	31:24	**Ctb column position of tile column 3** Format: U8
	23:16	**Ctb column position of tile column 2** Format: U8
	15:8	**Ctb column position of tile column 1** Format: U8
	7:0	**Ctb column position of tile column 0** Format: U8
3	31:24	**Ctb column position of tile column 7** Format: U8
	23:16	**Ctb column position of tile column 6** Format: U8
	15:8	**Ctb column position of tile column 5** Format: U8
	7:0	**Ctb column position of tile column 4** Format: U8
4	31:24	**Ctb column position of tile column 11** Format: U8
	23:16	**Ctb column position of tile column 10** Format: U8
	15:8	**Ctb column position of tile column 9** Format: U8
	7:0	**Ctb column position of tile column 8** Format: U8
5	31:24	**Ctb column position of tile column 15** Format: U8
	23:16	**Ctb column position of tile column 14** Format: U8
	15:8	**Ctb column position of tile column 13** Format: U8
	7:0	**Ctb column position of tile column 12** Format: U8

HCP_TILE_STATE		
6	31:24	**Ctb column position of tile column 19** Format: U8
	23:16	**Ctb column position of tile column 18** Format: U8
	15:8	**Ctb column position of tile column 17** Format: U8
	7:0	**Ctb column position of tile column 16** Format: U8
7	31:24	**Ctb row position of tile row 3** Format: U8
	23:16	**Ctb row position of tile row 2** Format: U8
	15:8	**Ctb row position of tile row 1** Format: U8
	7:0	**Ctb row position of tile row 0** Format: U8
8	31:24	**Ctb row position of tile row 7** Format: U8
	23:16	**Ctb row position of tile row 6** Format: U8
	15:8	**Ctb row position of tile row 5** Format: U8
	7:0	**Ctb row position of tile row 4** Format: U8
9	31:24	**Ctb row position of tile row 11** Format: U8
	23:16	**Ctb row position of tile row 10** Format: U8
	15:8	**Ctb row position of tile row 9** Format: U8
	7:0	**Ctb row position of tile row 8** Format: U8
10	31:24	**Ctb row position of tile row 15** Format: U8

HCP_TILE_STATE		
	23:16	**Ctb row position of tile row 14** Format: U8
	15:8	**Ctb row position of tile row 13** Format: U8
	7:0	**Ctb row position of tile row 12** Format: U8
11	31:24	**Ctb row position of tile row 19** Format: U8
	23:16	**Ctb row position of tile row 18** Format: U8
	15:8	**Ctb row position of tile row 17** Format: U8
	7:0	**Ctb row position of tile row 16** Format: U8
12	31:16	**Reserved** Format: MBZ
	15:8	**Ctb row position of tile row 21** Format: U8
	7:0	**Ctb row position of tile row 20** Format: U8

HCP_WEIGHTOFFSET_STATE

<table>
<tr><th colspan="3">HCP_WEIGHTOFFSET_STATE</th></tr>
<tr><td colspan="3">Project: CHV, BSW
Source: VideoCS
Length Bias: 2</td></tr>
<tr><td colspan="3">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.</td></tr>
<tr><td colspan="3">This slice level command is issued in both the encoding and decoding processes, if the weighted_pred_flag or weighted_bipred_flag equals one. If zero, then this command is not issued.
Weight Prediction Values are provided in this command. Only Explicit Weight Prediction is supported in encoder.
For P-Slice, this command is issued only once together with HCP_REF_IDX_STATE Command for L0 list. For B-Slice, this command can be issued up to two times together with HCP_REF_IDX_STATE Command, one for L0 list and one for L1 list.</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td>Command Type
Default Value: 3h PARALLEL_VIDEO_PIPE
Format: OpCode</td></tr>
<tr><td></td><td>28:27</td><td>Pipeline Type
Default Value: 2h
Format: OpCode</td></tr>
<tr><td></td><td>26:23</td><td>Media Instruction Opcode
Default Value: 7h Codec/Engine Name
Format: OpCode
Codec/Engine Name = HCP = 7h</td></tr>
<tr><td></td><td>22:16</td><td>Media Instruction Command
Default Value: 13h HCP_WEIGHTOFFSET_STATE
Format: OpCode</td></tr>
<tr><td></td><td>15:12</td><td>Reserved
Format: MBZ</td></tr>
<tr><td></td><td>11:0</td><td>Dword Length
Format: =n
(Excludes Dwords 0, 1).
<table><tr><th>Value</th><th>Name</th><th>Project</th></tr><tr><td>20h</td><td>32</td><td></td></tr></table></td></tr>
<tr><td>1</td><td>31:1</td><td>Reserved</td></tr>
</table>

HCP_WEIGHTOFFSET_STATE

DWord	Bit	Description
		Format: MBZ
	0	**RefPicListNum** Format: U1 Value 0: Reference Picture List 0 Value 1: Reference Picture List 1
2..17	31:16	**Reserved**
	15:8	**luma_offset_lX[i]** Format: U8 Where X is the RefPicListNum and i is the list entry number 0 through 15. DW2 corresponds to i=0, DW17 corresponds to i=15. Valid only if explicit weighted prediction for luma is enabled, otherwise must be zero.
	7:0	**delta_luma_weight_lX[i]** Format: U8 Where X is the RefPicListNum and i is the list entry number 0 through 15. DW2 corresponds to i=0, DW17 corresponds to i=15. Valid only if explicit weighted prediction for luma is enabled, otherwise must be zero.
18..33	31:24	**ChromaOffsetLX [i][1]** Format: U8 Where X is the RefPicListNum and i is the list entry number 0 through 15. DW2 corresponds to i=0, DW17 corresponds to i=15. Valid only if explicit weighted prediction for chroma is enabled, otherwise must be zero.
	23:16	**delta_chroma_weight_lX[i][1]** Format: U8 Where X is the RefPicListNum and i is the list entry number 0 through 15. DW2 corresponds to i=0, DW17 corresponds to i=15. Valid only if explicit weighted prediction for chroma is enabled, otherwise must be zero.
	15:8	**ChromaOffsetLX[i][0]** Format: U8 Where X is the RefPicListNum and i is the list entry number 0 through 15. DW2 corresponds to i=0, DW17 corresponds to i=15. Valid only if explicit weighted prediction for chroma is enabled, otherwise must be zero.

<table>
<tr><th colspan="3">HCP_WEIGHTOFFSET_STATE</th></tr>
<tr><td></td><td>7:0</td><td>delta_chroma_weight_lX[i][0]
<table><tr><td>Format:</td><td>U8</td></tr></table>
<table><tr><td>Where X is the RefPicListNum and i is the list entry number 0 through 15. DW2 corresponds to i=0, DW17 corresponds to i=15.</td></tr>
<tr><td>Valid only if explicit weighted prediction for chroma is enabled, otherwise must be zero.</td></tr></table>
</td></tr>
</table>

HI8DS Render Target Write MSD

MSD_RTW_HI8DS - HI8DS Render Target Write MSD

Project:	CHV, BSW
Source:	Render Cache DataPort
Length Bias:	1
Family:	Other
Group:	Render Target R/W

DWord	Bit	Description
0	31	**Reserved**
	30	**Message Precision Subtype** Default Value: 0h Project: All Format: Opcode Full precision data message
	29	**Reserved**
	28:25	**Message Length** Project: All Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Project: All Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the 2-register header.
	18	**Reserved**
	17:14	**Message Type** Default Value: 0Ch Project: All Format: Opcode Render Target Write message

MSD_RTW_HI8DS - HI8DS Render Target Write MSD

13 — Reserved

Project:	CHV, BSW
Format:	MBZ

Ignored

12 — Last Render Target Select

Project:	All
Format:	Enable

This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.

Programming Notes
When a pixel shader has render target writes at finer granularity than the dispatch rate, last render target write to a null surface must be present at the dispatch rate with this bit set. In particular, if a kernel is dispatched at pixel rate and it only writes to render targets at sample-rate, it must include a pixel-rate render target write to a null surface with Last Render Target Select bit enabled.

11 — Slot Group Select

Project:	All
Format:	MDC_RT_SGS [CHV, BSW]

This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.

10:8 — Render Target Message Subtype

Default Value:	3h
Project:	All
Format:	Opcode

SIMD8 dual source message. Use slots [15:8] for pixel enables, X/Y addresses, and oMask.

Programming Notes
The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [31:24] are referenced instead of [15:8].

7:0 — Binding Table Index

Project:	All
Format:	MDC_BTS [CHV, BSW]

Specifies the Binding Table Index for the message

HUC_CFG_STATE

HUC_CFG_STATE

Project:	CHV, BSW
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_CFG_STATE command is used to force the P24C (MinuteIA) into a reset condition as well as forcing it out of a reset condition. This command is not normally required since the hardware will handle placing the P24C into a reset condition and releasing it from reset, but there may be conditions that require a forced reset.

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: 3h HUC_CFG_STATE Format: OpCode
	15:12	**Reserved**
	11:0	**Dword Length** Format: U12 (Excludes Dwords 0, 1). Value: 0h — Name: (none)
1	31:1	**Reserved**
	0	**P24C (MinuteIA)** Format: U1 Value 0 — Name: Normal operation — Description: No reset. Value 1 — Name: Force reset — Description: (none)

HUC_DMEM_STATE

HUC_DMEM_STATE

Project: CHV, BSW
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_DMEM_STATE command is used to fetch the HUC data from the graphics memory and load it into the HUC 96KB L2 storage RAM. The HUC_DMEM_STATE specifies the data source base address in graphics memory.
When the HUC_DMEM_STATE command is received, the data is loaded by the HUC DMA into the 96KB L2 storage RAM at the location provided in the HUC_DMEM_STATE command. This command also specifies the length of the data, which is specified in bytes but must be in increments of 64 byte cache lines.

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: 2h HUC_DMEM_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: U12 (Excludes Dwords 0, 1). Value: 4h; Name: (blank)
1..2	63:0	**HUC Data Source Base Address** Format: SplitBaseAddress64ByteAligned Specifies the 64 byte aligned HUC data source base address in graphics memory.

HUC_DMEM_STATE		
3	31:0	**HUC Data Source** Project: CHV, BSW Format: MemoryAddressAttributes [CHV, BSW]
4	31:17	**Reserved** Format: MBZ
	16:6	**HUC Data Destination Base Address** Format: GraphicsAddress[16:6] Specifies the HUC Data destination base address in the L2 storage RAM. The base address is 64 byte cache aligned.
	5:0	**Reserved** Format: MBZ
5	31:17	**Reserved** Format: MBZ
	16:6	**HUC Data Length** Format: GraphicsAddress[16:6] Specifies the length in bytes of the HUC Data. The length must be in increments of 64 byte cache lines.
	5:0	**Reserved** Format: MBZ

HUC_IMEM_STATE

HUC_IMEM_STATE

Project:	CHV, BSW
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_IMEM_STATE command is used to fetch the HUC firmware from the WOPCM region and load it into the HUC 96KB L2 storage RAM. The HUC_IMEM_STATE specifies the firmware's offset in WOPCM which is a cache line aligned 32-bit offset address.
When the HUC_IMEM_STATE command is received, the hardware confirms that the code has been successfully authenticated by checking the VCR provided authentication successful signal. If this signal is asserted, the firmware is loaded by the HUC DMA into the 96KB L2 storage RAM. Once the firmware is loaded, the VALID IMEM LOADED bit in the HUC_STATUS2 register is asserted high in the. If the authentication signal is not asserted, the DMA aborts the HUC_IMEM_STATE command, the firmware is not loaded, and the VALID IMEM LOADED bit remains low.

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: 1h HUC_IMEM_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: U12 (Excludes Dwords 0, 1). Value: 3h — Name: (none)

HUC_IMEM_STATE		
1	31:0	**Reserved** Format: MBZ
2	31:0	**Reserved** Format: MBZ
3	31:0	**Reserved** Format: MBZ
4	31:8	**Reserved** Format: MBZ
	7:0	**HUC Firmware Descriptor** Format: U8 This field specifies 1 of 255 firmware descriptors which describe which firmware is be loaded in the L2 storage RAM. If the firmware descriptor is set to zero, the HUC will not load the firmware.

Value	Name	Description
0		Illegal
[1,255]		Firmware Descriptor

HUC_IND_OBJ_BASE_ADDR_STATE

HUC_IND_OBJ_BASE_ADDR_STATE

Project:	CHV, BSW
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_IND_OBJ_BASE_ADDR_STATE command is used to define the indirect object base address of the stream in graphics memory. This is a frame level 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: Bh Codec/Engine Name Format: OpCode Codec/Engine Name = HUC = Bh
	22:16	**Media Instruction Command** Default Value: 5h HUC_IND_OBJ_BASE_ADDR_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: U12 (Excludes Dwords 0, 1). Value: 9h — Name: (blank)
1..2	63:0	**HUC Indirect Stream In ObjectBase Address** Format: SplitBaseAddress4KByteAligned Specifies the 4K-byte aligned memory base address for the read-only indirect data object pointed in the HUC_STREAM_OBJECT command for fetching (reading) the bit stream data.

HUC_IND_OBJ_BASE_ADDR_STATE		
3	31:0	**HUC Indirect Stream In ObjectBase Attributes** Format: MemoryAddressAttributes [CHV, BSW]
4..5	63:0	**HUC Indirect Stream In ObjectAccess Upper Bound** Format: SplitBaseAddress4KByteAligned This field specifies the 4K-byte aligned maximum memory address access by the indirect data object in the HUC_STREAM_OBJECT command for the input bit stream. Indirect data accessed at this address or greater will cause the HUC to stop issuing requests to the GAC. Setting this field to 0 will cause this range to be ignored by the HUC.
6..7	63:0	**HUC Indirect Stream Out ObjectBase Address** Format: SplitBaseAddress4KByteAligned Specifies the 4K-byte aligned memory base address for the indirect data object pointed in the HUC_STREAM_OBJECT command for writing the bit stream data.
8	31:0	**HUC Indirect Stream Out ObjectBase Attributes** Format: MemoryAddressAttributes [CHV, BSW]
9..10	63:0	**HUC Indirect Stream Out ObjectAccess Upper Bound** Format: SplitBaseAddress4KByteAligned This field specifies the 4K-byte aligned maximum memory address access by the indirect data object in the HUC_STREAM_OBJECT command for the output bit stream. Indirect data accessed at this address or greater will cause the HUC to stop issuing requests to the GAC. Setting this field to 0 will cause this range to be ignored by the HUC.

HUC_PIPE_MODE_SELECT

HUC_PIPE_MODE_SELECT

Project:	CHV, BSW
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: (blank)
1	31:24	**Reserved**
	23:11	**Reserved** Format: MBZ
	9:5	**Reserved** Format: MBZ

HUC_PIPE_MODE_SELECT		
	4	**Indirect Stream Out Enable** Format: Enable Enables the bitstream to be written out to memory immediately following the decryption of the bit stream. 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** / **Project** 0 / Disable / CHV, BSW

HUC_START

HUC_START

Project:	CHV, BSW
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.

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: 21h HUC_START Format: OpCode
	15:12	**Reserved**
	11:0	**Dword Length** Format: U12 (Excludes Dwords 0, 1). Value: 0h; Name: (blank)
1	31:3	**Reserved**
	2:1	**Set Verbose Debug Bits** Format: U2
	0	**LastStreamObject** Format: U1 Value 1 — LastStreamObject — Last stream object in the workload. Value 0 — NotLastStreamObject — Not the last stream object in the workload.

HUC_STREAM_OBJECT

HUC_STREAM_OBJECT

Project:	CHV, BSW
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: U12 (Excludes Dwords 0, 1). Value: 3h — Name: (none)
1	31:0	**Indirect Stream In Data Length** Format: U32 Specifies the length in bytes of the bit stream input data.

<table>
<tr><th colspan="3">HUC_STREAM_OBJECT</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>[0,268435455]</td><td>Data_Length_with_28_bits_only</td><td>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.</td></tr>
</table>
</td></tr>
<tr><td>2</td><td>31</td><td>
HuC Processing
<table><tr><td>Format:</td><td>Disable</td></tr></table>
Disables the HEVC Decoder CABAC engine to prevent it from starting while the HuC is processing. Must be set to "1" for HUC processing so that teh stream is directed to the HuC and not the CABAC engine.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>Disable</td></tr>
</table>
</td></tr>
<tr><td></td><td>30:29</td><td>
Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table>
</td></tr>
<tr><td></td><td>28:0</td><td>
Indirect Stream In Start Address
<table><tr><td>Format:</td><td>U29</td></tr></table>
Specifies the byte-aligned graphics memory starting address of the input bit stream relative to the HUC Indirect Stream In ObjectBase Address [31:12].
</td></tr>
<tr><td>3</td><td>31:29</td><td>
Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table>
</td></tr>
<tr><td></td><td>28:0</td><td>
Indirect Stream Out Start Address
<table><tr><td>Format:</td><td>U29</td></tr></table>
Specifies the byte-aligned graphics memory starting address of the output bit stream relative to the HUC Indirect Stream Out ObjectBase Address [31:12].
</td></tr>
<tr><td>4</td><td>31:30</td><td>
Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table>
</td></tr>
<tr><td></td><td>29</td><td>
HuC Bitstream Enable
<table><tr><td>Format:</td><td>Enable</td></tr></table>
Enables the bitstream to be sent to the HuC
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0h</td><td>Disable</td></tr>
<tr><td>1h</td><td>Enable</td></tr>
</table>
</td></tr>
</table>

HUC_STREAM_OBJECT

	Bits	Field
	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	**Stream Out** Format: Enable Enables the serpent encrypted stream output. Value / Name / Description: 0 — Disable — Disable the serpent encrypted stream output. 1 — Enable — Enable the serpent encrypted stream output.
	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

HUC_VIRTUAL_ADDR_STATE

HUC_VIRTUAL_ADDR_STATE

Project:	CHV, BSW
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_VIRTUAL_ADDR_STATE command is used to define the 48-bit HUC surface base address for each of the 16 regions.

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: 4h HUC_VIRTUAL_ADDR_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: U12 (Excludes Dwords 0, 1). Value: 2Fh — Name:
1..48	95:64	**HUC Surface (VirtualAddrRegion[0-15])** Format: MemoryAddressAttributes [CHV, BSW]
	63:0	**HUC Surface Base Address (VirtualAddrRegion[0-15])** Format: SplitBaseAddress4KByteAligned HUC surface base address for each virtual address region [0-15].

Illegal

illegal - Illegal

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The Illegal Opcode Exception Enable flag in cr0.1 is normally set so the normal processing of an illegal opcode is to transfer control to the System Routine. Instruction dispatch treats any unused 8-bit opcode (including bit 7 of the instruction, reserved for future opcode expansion) as if it is the illegal opcode. The illegal opcode is zero because that byte value is more likely than most to be read via a wayward instruction pointer. The illegal instruction is an instruction only in the same way that a NULL pointer in software is a pointer. Both are special values indicating invalid instances.

Format: illegal

Restriction
The illegal instruction takes no instruction options.

Syntax
illegal

Pseudocode
{ Set the Illegal Opcode Exception Status bit in cr0.1. if (Illegal Opcode Exception Enable is set in cr0.1) { Transfer control to the System Routine (return address to AIP, IP = SIP). } }

Predication	Conditional Modifier	Saturation	Source Modifier
N	N	N	N

DWord	Bit	Description
0..3	127:7	**Reserved** Format: MBZ
	6:0	**Opcode** Format: EU_OPCODE [CHV, BSW]

Integer Subtraction with Borrow

subb - Integer Subtraction with Borrow

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The subb instruction performs component-wise subtraction of src0 and src1 and stores the results in dst, it also stores the borrow into acc. If the operation produces a borrow (src0 < src1), write 0x00000001 to acc, else write 0x00000000 to acc.

Format: [(pred)] subb[.cmod] (exec_size) dst src0 src1

Restriction
AccWrEn is required. The accumulator is an implicit destination and thus cannot be an explicit destination operand.

Syntax
[(pred)] subb[.cmod] (exec_size) reg reg reg [(pred)] subb[.cmod] (exec_size) reg reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = src0.chan[n] - src1.chan[n]; acc.chan[n] = borrow(src.chan[n] - src1.chan[n]); } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	Y	N

Src Types	Dst Types
UD	UD

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Join

join - Join

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The join instruction makes the inactive channels active at the join IP if those channels are predicated. Any deactivated channels due to a goto instruction match the join IP are activated (qualified with predicates at join). If no IP is matched at this join, the program goes to the next IP with the active channels which followed the program path up to the join instruction. If no active channels are present after executing the join instruction, the program jumps to the offset specified by JIP instead of next IP. The join instruction is used in conjunction with a goto instruction. The join activates channels that are deactivated by the goto instruction. See the goto instruction for the deactivation rules. The goto and join instructions enable unstructured program control flow. These instructions must be used with additional care where dangling channels can result without proper compiler checks, meaning that it is expected that programs will navigate through these paths to reactivate the channels. Hardware does not provide native checks or reconvergence. The following table describes the 32-bit JIP. In GEN binary, JIP is at location src1 and must be of type D (signed DWord integer). JIP must be an immediate operand and is a signed 32-bit number. This value is added to IP pre-increment. If SPF is ON, none of the PcIP are updated.

Format: [(pred)] join (exec_size) JIP

Programming Notes
An index of 0 does nothing, continuing execution with the next instruction.
An index of -16 (if the jmpi instruction is in native format) or -8 (if the jmpi instruction is in compact format) is an infinite loop on the jmpi instruction.

Restriction
The {NoMask} instruction option must be specified.
The index data type must be D (Signed DWord Integer).

Syntax
[(pred)] join (exec_size) imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { // for the predicated channels and the remaining channels PcIP[n] = IP + 1; } } if (PcIP != (IP + 1)) { // for all channels when no channels are activated and no other active channels Jump(IP + JIP); }

Errata	Description
	A join instruction must not be followed by any instruction requiring register indirect access on source operands.

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

DWord	Bit	Description
0..3	127:96	**JIP** Project: CHV, BSW Format: S31 Jump Target Offset. The relative offset in bytes if a jump is taken for the instruction.
	95	**Source 0 Address Immediate [9] Sign Bit** Project: CHV, BSW
	94:91	**Src1.SrcType** Project: CHV, BSW Format: SrcType [CHV, BSW]
	90:89	**Src1.RegFile** Project: CHV, BSW Format: RegFile [CHV, BSW]
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align16') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN16 [CHV, BSW]
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN1 [CHV, BSW]
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Jump Indexed

jmpi - Jump Indexed

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

Description
The jmpi instruction redirects program execution to an index offset relative to the post-incremented instruction pointer. The index is a signed integer value, with positive or zero integers for forward jumps, and negative integers for backward jumps. Note: Unlike other flow control instructions, the offset used by jmpi is relative to the incremented instruction pointer rather than the IP value for the instruction itself. In GEN binary, index is at location src1. The ip register must be put (for example, by the assembler) at the dst and src0 locations. Predication is allowed to provide conditional jump with a scalar condition. As the execution size is 1, the first channel of PMASK (flags post prediction control and negate) is used to determine whether the jump is taken or not. If the condition is false, the jump is not taken and execution continues with the next instruction.
Format: [(pred)] jmpi (1) index {NoMask}

Programming Notes
An index of 0 does nothing, continuing execution with the next instruction.
An index of -16 (if the jmpi instruction is in native format) or -8 (if the jmpi instruction is in compact format) is an infinite loop on the jmpi instruction.

Restriction
The execution size must be 1.
The {NoMask} instruction option must be specified.
The index data type must be D (Signed DWord Integer).
QtrCtrl must not be used for jmpi instruction.

Syntax
[(pred)] jmpi (1) reg32 {NoMask} [(pred)] jmpi (1) imm32 {NoMask}

Pseudocode
Evaluate(WrEn); if (WrEn != 0) { Jump(IP + 1 + index); // IP + 1 is a pseudocode idiom for the IP of the following instruction. }

Errata	Description
	A jmpi instruction must not be followed by any instruction requiring register indirect access on source operands.

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

jmpi - Jump Indexed

Src Types
D

DWord	Bit	Description
0..3	127:96	**JIP** Project: CHV, BSW Format: S31 Jump Target Offset. The relative offset in bytes if a jump is taken for the instruction.
	95	**Source 0 Address Immediate [9] Sign Bit** Project: CHV, BSW
	94:91	**Src1.SrcType** Project: CHV, BSW Format: SrcType [CHV, BSW]
	90:89	**Src1.RegFile** Project: CHV, BSW Format: RegFile [CHV, BSW]
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align16') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN16 [CHV, BSW]
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN1 [CHV, BSW]
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Leading Zero Detection

lzd - Leading Zero Detection

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The lzd instruction counts component-wise the leading zeros from src0 and stores the resulting counts in dst. If src0 is zero, store 32 in dst.

Format: [(pred)] lzd[.cmod] (exec_size) dst src0

Restriction
Accumulator cannot be destination, implicit or explicit.

Syntax
[(pred)] lzd[.cmod] (exec_size) reg reg [(pred)] lzd[.cmod] (exec_size) reg reg

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { UD udScalar = src0.chan[n]; UD cnt = 0; while ((udScalar & (1 « 31)) == 0 && cnt != 32) { cnt ++; udScalar = udScalar « 1; } dst.chan[n] = cnt; } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types
D, UD	UD

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32 [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Line

line - Line

Project:	CHV, BSW
Source:	Eulsa
Length Bias:	4

The line instruction computes a component-wise line equation (v = p * u + q where u, v are vectors and p, q are scalars) of src0 and src1 and stores the results in dst. src1 is the input vector u. src0 provides input scalars p and q, where p is the scalar value based on the region description of src0 and q is the scalar value implied from src0 region. Specifically, q is the fourth component of the 4-tuple (128-bit aligned) that p belongs to.

Format: [(pred)] line[.cmod] (exec_size) dst src0 src1

Restriction
This is a specialized instruction that only supports an execution size (ExecSize) of 8 or 16.
The src0 region must be a replicated scalar (with HorzStride == VertStride == 0).
src0 must specify .0 or .4 as the subregister number, corresponding to a subregister byte offset of 0 or 16.
Source operands cannot be accumulators.

Syntax
[(pred)] line[.cmod] (exec_size) reg reg reg [(pred)] line[.cmod] (exec_size) reg reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { dwP = src0.RegNum.SubRegNum[bits4:2]; // A DWord-aligned scalar. dwQ = src0.RegNum.(SubRegNum[bit4] | 0x8); // Fourth component. if (WrEn.chan[n]) { dst.chan[n] = dwP * src1.chan[n] + dwQ; } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types
F	F

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Linear Interpolation

lrp - Linear Interpolation

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The lrp instruction takes component-wise multiplication of src0 and src1, and adds the result to the component-wise multiplication of src2 and (1 - src0), and then stores the final results in dst.

Format: [(pred)] lrp[.cmod] (exec_size) dst src0 src1 src2

Restriction
The vertical stride (VertStride) is overloaded to 4 in HW for 3-source instructions.
The overflow conditional modifier (.o) is not allowed.
No explicit accumulator access because this is a three-source instruction. AccWrEn is allowed for implicitly updating the accumulator.
All three-source instructions have certain restrictions, described in Instruction Formats [CHV, BSW].

Syntax
[(pred)] lrp[.cmod] (exec_size) reg reg reg

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = src1.chan[n] * src0.chan[n] + src2.chan[n] * (1.0 - src0.chan[n]); } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	Y	Y

Src Types	Dst Types
F	F

DWord	Bit	Description
0..3	127:126	**Reserved** Format: MBZ
	125:106	**Source 2** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC [CHV, BSW]
	105	**Reserved** Format: MBZ
	104:85	**Source 1** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC [CHV, BSW]
	84	**Reserved** Format: MBZ

lrp - Linear Interpolation

	83:64	**Source 0** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC [CHV, BSW]
	63:56	**Destination Register Number** Format: DstRegNum [CHV, BSW]
	55:53	**Destination Subregister Number** Format: DstSubRegNum[2:0]
	52:49	**Destination Channel Enable** Format: ChanEn[4] Four channel enables are defined for controlling which channels are written into the destination region. These channel mask bits are applied in a modulo-four manner to all ExecSize channels. There is 1-bit Channel Enable for each channel within the group of 4. If the bit is cleared, the write for the corresponding channel is disabled. If the bit is set, the write is enabled. Mnemonics for the bit being set for the group of 4 are x, y, z, and w, respectively, where x corresponds to Channel 0 in the group and w corresponds to channel 3 in the group
	48:42	**Reserved**
	41:40	**Source 2 Modifier** Exists If: //([Property[Source Modifier]=='true') Format: SrcMod [CHV, BSW]
	39:38	**Source 1 Modifier** Exists If: //([Property[Source Modifier]=='true') Format: SrcMod [CHV, BSW]
	41:36	**Reserved** Exists If: //([Property[Source Modifier]=='false') Format: MBZ
	37:36	**Source 0 Modifier** Exists If: //([Property[Source Modifier]=='true') Format: SrcMod [CHV, BSW]
	35	**Reserved**
	34	**Reserved** Format: MBZ
	33	**Flag Subregister Number** This field contains the flag subregister number for instructions with a non-zero Conditional Modifier.
	32	**Reserved** Format: MBZ
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

LO8DS Render Target Write MSD

MSD_RTW_LO8DS - LO8DS Render Target Write MSD

Project:	CHV, BSW
Source:	Render Cache DataPort
Length Bias:	1
Family:	Other
Group:	Render Target R/W

DWord	Bit	Description
0	31	**Reserved**
	30	**Message Precision Subtype** Default Value: 0h Project: All Format: Opcode Full precision data message
	29	**Reserved**
	28:25	**Message Length** Project: All Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Project: All Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the 2-register header.
	18	**Reserved**
	17:14	**Message Type** Default Value: 0Ch Project: All Format: Opcode Render Target Write message

MSD_RTW_LO8DS - LO8DS Render Target Write MSD

	13	**Reserved**
		Project: CHV, BSW
		Format: MBZ
		Ignored
	12	**Last Render Target Select**
		Project: All
		Format: Enable
		This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.
		Programming Notes When a pixel shader has render target writes at finer granularity than the dispatch rate, last render target write to a null surface must be present at the dispatch rate with this bit set. In particular, if a kernel is dispatched at pixel rate and it only writes to render targets at sample-rate, it must include a pixel-rate render target write to a null surface with Last Render Target Select bit enabled.
	11	**Slot Group Select**
		Project: All
		Format: MDC_RT_SGS [CHV, BSW]
		This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
	10:8	**Render Target Message Subtype**
		Default Value: 2h
		Project: All
		Format: Opcode
		SIMD8 dual source message. Use slots [7:0] for pixel enables, X/Y addresses, and oMask.
		Programming Notes The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [23:16] are referenced instead of [7:0].
	7:0	**Binding Table Index**
		Project: All
		Format: MDC_BTS [CHV, BSW]
		Specifies the Binding Table Index for the message

Logic And

and - Logic And

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The and instruction performs component-wise logic AND operation between src0 and src1 and stores the results in dst. Register source operands can use source modifiers: Any source modifier is logical, optionally changing a source value s to ~s (inverting all source bits). This capability allows expressions like a AND (NOT b) to be calculated with one instruction. This operation does not produce sign or overflow conditions. Only the .e/.z or .ne/.nz conditional modifiers should be used.

Format: Source modifier is not allowed if source is an accumulator.

Restriction
Source modifier is not allowed if source is an accumulator.

Syntax
[(pred)] and[.cmod] (exec_size) reg reg reg [(pred)] and[.cmod] (exec_size) reg reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = src0.chan[n] & src1.chan[n]; } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	N	N

Src Types	Dst Types	Project
*B,*W,*D	*B,*W,*D	
*W,*D,*Q	*W,*D,*Q	CHV, BSW

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Logic Not

not - Logic Not

Project:	CHV, BSW
Source:	Eulsa
Length Bias:	4

Description
The not instruction performs logical NOT operation (or one's complement) of src0 and storing the results in dst. This operation does not produce sign or overflow conditions. Only the .e/.z or .ne/.nz conditional modifiers should be used.
A register source operand can use a source modifier: Any source modifier is logical, optionally changing a source value s to ~s (inverting all source bits). Such a source modifier is not particularly useful with the not instruction, as it changes the effect of not to just copying bits.
Format: [(pred)] not[.cmod] (exec_size) dst src0

Restriction
Source modifier is not allowed if source is an accumulator.

Syntax
[(pred)] not[.cmod] (exec_size) reg reg [(pred)] not[.cmod] (exec_size) reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = ~ src0.chan[n]; } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	Y

Src Types	Dst Types	Project
*B,*W,*D	*B,*W,*D	
*W,*D,*Q	*W,*D,*Q	CHV, BSW

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32 [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Logic Or

or - Logic Or

Project:	CHV, BSW
Source:	Eulsa
Length Bias:	4

Description
The or instruction performs component-wise logic OR operation between src0 and src1 and stores the results in dst. This operation does not produce sign or overflow conditions. Only the .e/.z or .ne/.nz conditional modifiers should be used.
Register source operands can use source modifiers: Any source modifier is logical, optionally changing a source value s to ~s (inverting all source bits). This capability allows expressions like a OR (NOT b) to be calculated with one instruction.
Format: [(pred)] or[.cmod] (exec_size) dst src0 src1

Restriction
Source modifier is not allowed if source is an accumulator.
Syntax
[(pred)] or[.cmod] (exec_size) reg reg reg [(pred)] or[.cmod] (exec_size) reg reg imm32
Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = src0.chan[n] | src1.chan[n]; } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	Y

Src Types	Dst Types	Project
*B,*W,*D	*B,*W,*D	
*W,*D,*Q	*W,*D,*Q	CHV, BSW

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Logic Xor

xor - Logic Xor

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

Description
The xor instruction performs component-wise logic XOR operation between src0 and src1 and stores the results in dst. This operation does not produce sign or overflow conditions. Only the .e/.z or .ne/.nz conditional modifiers should be used.
Register source operands can use source modifiers: Any source modifier is logical, optionally changing a source value s to ~s (inverting all source bits). This capability allows expressions like a XOR (NOT b) to be calculated with one instruction.
Format: [(pred)] xor[.cmod] (exec_size) dst src0 src1

Restriction
Source modifier is not allowed if source is an accumulator.

Syntax
[(pred)] xor[.cmod] (exec_size) reg reg reg [(pred)] xor[.cmod] (exec_size) reg reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = src0.chan[n] ^ src1.chan[n];}}

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	Y

Src Types	Dst Types	Project
*B,*W,*D	*B,*W,*D	
*W,*D,*Q	*W,*D,*Q	CHV, BSW

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

MEDIA_CURBE_LOAD

MEDIA_CURBE_LOAD		
Project: CHV, BSW Source: RenderCS Length Bias: 2		
Workaround: See "GPGPU Command Workarounds" section for additional programming constraints for this command.		
DWord	**Bit**	**Description**
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 MEDIA_CURBE_LOAD Format: OpCode
	23:16	**SubOpcode** Default Value: 1h MEDIA_CURBE_LOAD SubOp Format: OpCode
	15:0	**DWord Length** Project: All Format: =n Total Length - 2 Value / Name / Description: 2h / DWORD_COUNT_n **[Default]** / Excludes DWord (0,1)
1	31:0	**Reserved**
2	31:17	**Reserved**
	16:0	**CURBE Total Data Length** Project: All Format: U17 In Bytes Description: This field provides the length in bytes of the CURBE data. This field must have the same alignment as the Curbe Object Data Start Address.As the CURBE data are sent directly to ROB, range is limited to CURBE Allocation Size. This field must be 64-byte aligned.

MEDIA_CURBE_LOAD

3	31:0	**CURBE Data Start Address**

Project:	All
Format:	DynamicStateOffset[31:0] CURBE

Description	Project
Specifies the 64-byte aligned address of the CURBE data. This pointer is relative to the **Dynamics Base Address**.	CHV, BSW

Value	Name
[0, FFFFFFFFh]	

MEDIA_INTERFACE_DESCRIPTOR_LOAD

MEDIA_INTERFACE_DESCRIPTOR_LOAD

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

A Media_State_Flush should be used before this command to ensure that the temporary Interface Descriptor storage is cleared.

Workaround: See "GPGPU Command Workarounds" section for additional programming constraints for this command.

DWord	Bit	Description
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 MEDIA_INTERFACE_DESCRIPTOR_LOAD Format: OpCode
	23:16	**SubOpcode** Default Value: 2h MEDIA_INTERFACE_DESCRIPTOR_LOAD SubOp Format: OpCode
	15:0	**DWord Length** Format: =n Total Length - 2 Value / Name / Description: 2h / DWORD_COUNT_n **[Default]** / Excludes DWord (0,1)
1	31:0	**Reserved**
2	31:17	**Reserved**
	16:0	**Interface Descriptor Total Length** Format: U17 In bytes This field provides the length in bytes of the Interface Descriptor data. This field must have the same alignment as the Interface Descriptor Data Start Address.It must be DQWord (32-byte) aligned. As the Interface Descriptor data are sent directly to ROB, range is limited to CURBE Allocation Size. Value / Name / Project: [32,2048] / [1,64] interface descriptor entries / CHV, BSW

MEDIA_INTERFACE_DESCRIPTOR_LOAD		
3	31:0	**Interface Descriptor Data Start Address**

Format:	DynamicStateOffset[31:0]INTERFACE_DESCRIPTOR_DATA

Description	Project
This bit specifies the 64-byte aligned address of the Interface Descriptor data. This pointer is relative to the Dynamics Base Address.	CHV, BSW

Value	Name
[0, FFFFFFFFh]	

MEDIA_OBJECT

MEDIA_OBJECT

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

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 Format: OpCode
	23:16	**Media Command Sub-Opcode** Default Value: 0h MEDIA_OBJECT SubOp Format: OpCode
	15:0	**DWord Length** Default Value: 4h DWORD_COUNT_n Project: CHV, BSW Format: =n Total Length - 2 Excludes DWords 0,1 **Generic Mode:** DWord Length = N+4, where N is in the range of [0,504]. The maximum is 504 DW (equivalent to 63 8-DW registers). 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** Project: CHV, BSW 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.

<table>
<tr><th colspan="3">MEDIA_OBJECT</th></tr>
<tr><td rowspan="8">2</td><td>31</td><td>Children Present<table><tr><td>Format:</td><td>Enable</td></tr></table>Indicates that the root thread may send spawn messages to spawn child threads and/or synchronized root threads. If Children Present is not set, TS signals VFE to dereference the URB handle immediately after it receives acknowledgement from TD that the thread is dispatched. If Children Present is set, the URB handle is forwarded to the root thread and serves as the return URB handle for the root thread. TS does not signal deference at the time of dispatch. TS signals URB handle deference only when it receives a resource dereference message from the thread. In order avoid deadlock, such dereference must be issued once and only once for each URB handle.</td></tr>
<tr><td>30:27</td><td>Reserved</td></tr>
<tr><td>26:25</td><td>Reserved</td></tr>
<tr><td>24</td><td>Thread Synchronization
This field when set indicates that the dispatch of the thread originated from this command is based on the "spawn root thread" message.<table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>No thread synchronization</td></tr><tr><td>1</td><td>Thread dispatch is synchronized by the 'spawn root thread' message</td></tr></table></td></tr>
<tr><td>23</td><td>Reserved</td></tr>
<tr><td>22</td><td>Force Destination<table><tr><td>Project:</td><td>CHV, BSW</td></tr></table>If set, bits 20:17 are used to determine the destination of this dispatch, if clear the destination will be chosen based on load.</td></tr>
<tr><td>21</td><td>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.<table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>Not using scoreboard</td></tr><tr><td>1</td><td>Using scoreboard</td></tr></table></td></tr>
<tr><td>20:19</td><td>Slice Destination Select<table><tr><td>Project:</td><td>CHV, BSW</td></tr></table>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.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>00b</td><td>Slice 0</td><td></td></tr><tr><td>01b</td><td>Slice 1</td><td>Cannot be used in products without a Slice 1.</td></tr><tr><td>10b</td><td>Slice 2</td><td>Cannot be used in products without a Slice 2.</td></tr><tr><td>11b</td><td>Reserved</td><td></td></tr></table></td></tr>
</table>

MEDIA_OBJECT

DWord	Bit	Description
	18:17	**SubSlice Destination Select** Project: CHV, BSW This field selects the SubSlice that this thread must be sent to. Ignored if **Force Destination** = 0

Value	Name	Project
11b	Reserved	CHV, BSW
10b	SubSlice 2	
01b	SubSlice 1	
00b	SubSlice 0	

DWord	Bit	Description
	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 496 DW. 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 and indirect data length rounded up to 8-DW aligned).
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.
This field specifies the 64-byte aligned address of the indirect data.

Value	Name
[0,512MB]	

Programming Notes
Bits 31:29 MBZ

DWord	Bit	Description
4	31:25	**Reserved**
	24:16	**Scoreboard Y** Project: Format: U9 This field provides the Y term of the scoreboard value of the current thread.
	15:9	**Reserved**

MEDIA_OBJECT		
	8:0	**Scoreboard X** Project: | Format: | U9 This field provides the X term of the scoreboard value of the current thread.
5	31:20	**Reserved** Project: | Format: | MBZ
	19:16	**Scoreboard Color** Project: | 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** Project: | 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..n	31:0	**Inline Data** Generic Mode: 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_OBJECT_GRPID

MEDIA_OBJECT_GRPID

Project:	CHV, BSW
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].

<table>
<tr><th colspan="3">MEDIA_OBJECT_GRPID</th></tr>
<tr><td></td><td></td><td>The maximum is 504 DW (equivalent to 63 8-DW registers). 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.</td></tr>
<tr><td>1</td><td>31:8</td><td>Reserved</td></tr>
<tr><td></td><td>7:6</td><td>Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>5:0</td><td>Interface Descriptor Offset
<table><tr><td>Format:</td><td>U6</td></tr></table>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.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,30]</td><td></td></tr></table></td></tr>
<tr><td>2</td><td>31:24</td><td>Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>23</td><td>End of Thread Group
This bit indicates that this dispatch is the last for the current thread group.</td></tr>
<tr><td></td><td>22</td><td>Force Destination
<table><tr><td>Project:</td><td>CHV, BSW</td></tr></table>If set, bits 20:17 are used to determine the destination of this dispatch, if clear the destination will be chosen based on load.</td></tr>
<tr><td></td><td>21</td><td>Use Scoreboard
<table><tr><td>Project:</td><td></td></tr></table>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.<table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>Not using scoreboard</td></tr><tr><td>1</td><td>Using scoreboard</td></tr></table></td></tr>
<tr><td></td><td>20:19</td><td>Slice Destination Select
This bit along with the Tslice destination select determines the slice that this thread must be sent to. Ignored if Force Destination = 0, or if product only has 1 slice.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>00b</td><td>Slice 0</td><td></td></tr><tr><td>01b</td><td>Slice 1</td><td>Cannot be used in products without a Slice 1.</td></tr><tr><td>10b</td><td>Slice 2</td><td>Cannot be used in products without a Slice 2.</td></tr><tr><td>11b</td><td>Reserved</td><td></td></tr></table></td></tr>
</table>

MEDIA_OBJECT_GRPID

DWord	Bit	Description
	18:17	**SubSlice Destination Select** Project: CHV, BSW This field selects the SubSlice that this thread must be sent to. Ignored if **Force Destination** = 0 Value | Name | Project 11b | Reserved | CHV, BSW 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 496 DW. 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 and indirect data length rounded up to 8-DW aligned).
3	31:0	**Indirect Data Start Address** Format: GraphicsAddress[31:0] Description | Project 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. | CHV, BSW Value | Name | Description [0-512MB] | | Bits 31:29 MBZ
4	31:25	**Reserved** Format: MBZ
	24:16	**Scoreboard Y** Project: 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** Project: — Format: U9 This field provides the X term of the scoreboard value of the current thread.
5	31:20	**Reserved** Project: — Format: MBZ
	19:16	**Scoreboard Color** Project: — 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** Project: — 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_OBJECT_PRT

MEDIA_OBJECT_PRT

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

command is for generating Persistent Root Thread for the media pipeline. It only supports loading of inline data but not indirect data. This command should be used for a root thread that might have to be present in the system for a period longer than the certain minimal context-switch interrupt latency. It has to honor the context interrupt signal to terminate upon request. It should also handle replay from theinterrupted point upon context restore (the same thread being dispatched more than once). In contrary, if a thread is not a Persistent Root Thread, if dispatched, it must run to completion. The command can be used in all VFE modes, except VLD mode.

For simplification, _PRT command has a fixed size of 16 DWORD

DWord	Bit	Description
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: 1h MEDIA_OBJECT_PRT Format: OpCode
	23:16	**SubOpcode** Default Value: 2h MEDIA_OBJECT_PRT SubOp Format: OpCode
	15:0	**DWord Length** Project: CHV, BSW Format: =n Total Length - 2 Note: Regardless of the mode, inline data must be present in this command. The command size must fit within 16 dwords.
1	31:6	**Reserved** Format: MBZ

DWord Length values (DWord 0, bits 15:0):

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

<table>
<tr><th colspan="3">MEDIA_OBJECT_PRT</th></tr>
<tr><td></td><td>5:0</td><td>Interface Descriptor Offset
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>U6</td></tr></table>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.</td></tr>
<tr><td>2</td><td>31</td><td>Children Present
<table><tr><td>Format:</td><td>Enable</td></tr></table>Indicates that the root thread may send spawn messages to spawn child threads and/or synchronized root threads. If Children Present is not set, TS signals VFE to dereference the URB handle immediately after it receives acknowledgement from TD that the thread is dispatched. If Children Present is set, the URB handle is forwarded to the root thread and serves as the return URB handle for the root thread. TS does not signal deference at the time of dispatch. TS signals URB handle deference only when it receives a resource dereference message from the thread. In order avoid deadlock, such de-reference must be issued once and only once for each URB handle.</td></tr>
<tr><td></td><td>30:24</td><td>Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>23</td><td>PRT_Fence Needed
<table><tr><td>Format:</td><td>Enable</td></tr></table>This field specifies that a PRT_Fence is generated after dispatching the thread associated with this MEDIA_OBJECT_PRT. The PRT_Fence prevents additional threads following this persistent root thread until a thread spawn message is sent. The PRT_Fence is generated on first dispatch of the persistent root, as well as on re-dispatches of the persistent root after context restore.</td></tr>
<tr><td></td><td>22</td><td>PRT_FenceType
This field specifies the type of fence the PRT thread uses. If this field is set to 0, the fence is set at the end of the root thread queue. It will block the dispatch of the next root thread, but allowed these root threads to be populated through VFE to the root thread queue in TS. If this field is set to 1, the fence is set at the entry of VFE, similar to the fence set by the MEDIA_STATE_FLUSH command. No more command can go into the media pipe until a thread spawn message is sent (by the PRT). This field is only valid when PRT_Fence Needed is set to 1. Otherwise, it is ignored by hardware.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Root thread queue</td><td>Root thread queue fence</td></tr><tr><td>1h</td><td>VFE state flush</td><td>VFE state flush fence</td></tr></table></td></tr>
<tr><td></td><td>21:0</td><td>Reserved</td></tr>
<tr><td>3</td><td>31:0</td><td>Reserved</td></tr>
<tr><td>4..15</td><td>31:0</td><td>Inline Data
<table><tr><td>Format:</td><td>U32</td></tr></table></td></tr>
</table>

MEDIA_OBJECT_WALKER

MEDIA_OBJECT_WALKER

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

DWord	Bit	Description
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: 1h MEDIA_OBJECT_WALKER Format: OpCode
	23:16	**SubOpcode** Default Value: 03h MEDIA_OBJECT_WALKER SubOp Format: OpCode
	15:0	**DWord Length** Default Value: 0Fh DWORD_COUNT_n Format: =n Total Length - 2 **Note:** If this field is greater than 15, it indicates that inline data is present. If present, inline data is common for all threads generated from this command, If this field is 15, it indicates that inline data is not present. It should be noted that unlike other media object command, inline data is optional for this command.
1	31:8	**Reserved**
	7:6	**Reserved** Format: Reserved
	5:0	**Interface Descriptor Offset** Project: CHV, BSW 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.
2	31	**Children Present** Format: Boolean

MEDIA_OBJECT_WALKER

DWord	Bit	Description
		Indicates that the root thread may send spawn messages to spawn child threads and/or synchronized root threads. If Children Present is not set, TS signals VFE to dereference the URB handle immediately after it receives acknowledgement from TD that the thread is dispatched. If Children Present is set, the URB handle is forwarded to the root thread and serves as the return URB handle for the root thread. TS does not signal deference at the time of dispatch. TS signals URB handle deference only when it receives a resource dereference message from the thread. *In order avoid deadlock, such dereference must be issued once and only once for each URB handle.*
	30:25	**Reserved** Format: MBZ
	24	**Thread Synchronization** This field when set indicates that the dispatch of the thread originated from this command is based on the "spawn root thread" message. **Value** / **Name**: 0 — No thread synchronization 1 — Thread dispatch is synchronized by the 'spawn root thread' message
	23:22	**Reserved** Format: MBZ
	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
	20:17	**Reserved** Format: MBZ
	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 496 DW. When both inline and indirect data are fetched for this command, the total size in 8-DW registers must be less than or equal to 63 (with both inline data length and indirect data length rounded up to 8-DW aligned).
3	31:0	**Indirect Data Start Address** Format: IndirectObjectOffset[31:0]

MEDIA_OBJECT_WALKER

		(continued)

Description	Project
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	CHV, BSW

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

DWord 4, Bits 31:0 — Reserved

Format:	MBZ

DWord 5, Bits 31:8 — Group ID Loop Select

Project:	CHV, BSW

This bit field chooses which of the nested loops of the walker are used to identify threads which share a group id and therefore a shared barrier and SLM. The programmer must ensure that each group will fit into a single subslice. When barriers are enabled every group must have the same number of threads matching the number specified in the Interface Descriptor.

Value	Name	Description
0		Groups are not created, barriers and SLM are not allocated
1		Each complete iteration of the Color loop defines a group, the group id is the concatenation of the Outer global, Inner global, Outer local, Mid local and Inner local loop execution counts.
2		Each complete iteration of the Inner local loop and Color loop defines a group, the group id is the concatenation of the Outer global loop to the Mid local loop execution counts.
3		Each complete iteration of the Mid local loop and lower loops defines a group, the group id is the concatenation of the Outer global loop to the Outer local loop execution counts.
4		Each complete iteration of the Outer local loop and lower loops defines a group, the group id is the concatenation of the Outer global loop and the Inner global loop execution counts.
5		Each complete iteration of the Inner global loop and lower loops defines a group, the group id is the Outer global loop execution count.

DWord 5, Bits 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. All threads generated by this walker command share the same dynamic mask. **Bit n (for n = 0...7):** Scoreboard n is dependent, where bit 0 maps to n = 0.

MEDIA_OBJECT_WALKER

6	31:29	**Reserved** Project: CHV, BSW
	28	**Reserved** Format: MBZ
	27:24	**Color Count Minus One** Format: U4 This field specifies the number of repeat of the inner most loop of the walker. Each repeated walk position is assigned with an incremental Color number. The Color number together with the X and Y position of the thread is used for dependency scoreboard control. **Usage Example:** This allows multiple sets of dependency threads to be dispatched.
	23:21	**Reserved** Format: MBZ
	20:16	**Middle Loop Extra Steps** Format: U5
	15:14	**Reserved** Format: MBZ
	13:12	**Local Mid-Loop Unit Y** Format: S1
	11:10	**Reserved** Format: MBZ
	9:8	**Mid-Loop Unit X** Format: S1
	7:0	**Reserved** Format: MBZ
7	31:26	**Reserved** Format: MBZ
	25:16	**Global Loop Exec Count** Format: U10
	15:10	**Reserved** Format: MBZ
	9:0	**Local Loop Exec Count** Format: U10
8	31:25	**Reserved** Format: MBZ
	24:16	**Block Resolution Y** Format: U9

MEDIA_OBJECT_WALKER		
		Vertical resolution of the local loop.
	15:9	**Reserved** Format: MBZ
	8:0	**Block Resolution X** Format: U9 Horizontal resolution of the local loop.
9	31:25	**Reserved** Format: MBZ
	24:16	**Local Start Y** Format: U9 Starting vertical position of the local loop.
	15:9	**Reserved** Format: MBZ
	8:0	**Local Start X** Format: U9 Starting horizontal position of the local loop.
10	31:25	**Reserved** Format: MBZ
	24:16	**Reserved** Project: CHV, BSW Format: MBZ
	15:9	**Reserved** Format: MBZ
	8:0	**Reserved** Project: CHV, BSW Format: MBZ
11	31:26	**Reserved** Format: MBZ
	25:16	**Local Outer Loop Stride Y** Format: S9 Vertical stride of the local outer loop, in 2's complement.
	15:10	**Reserved** Format: MBZ

MEDIA_OBJECT_WALKER		
	9:0	**Local Outer Loop Stride X** Format: S9 Horizontal stride of the local outer loop, in 2's complement.
12	31:26	**Reserved** Format: MBZ
	25:16	**Local Inner Loop Unit Y** Format: S9 Vertical stride of the local inner loop, in 2's complement.
	15:10	**Reserved** Format: MBZ
	9:0	**Local Inner Loop Unit X** Format: S9 Horizontal stride of the local inner loop, in 2's complement.
13	31:25	**Reserved** Format: MBZ
	24:16	**Global Resolution Y** Format: U9 Vertical resolution of the global loop.
	15:9	**Reserved** Format: MBZ
	8:0	**Global Resolution X** Format: U9 Horizontal resolution of the global loop.
14	31:26	**Reserved** Format: MBZ
	25:16	**Global Start Y** Format: S9 Starting vertical location of the global loop, in 2's complement.
	15:10	**Reserved** Format: MBZ
	9:0	**Global Start X** Format: S9

MEDIA_OBJECT_WALKER		
		Starting horizontal location of the global loop, in 2's complement.
15	31:26	**Reserved** Format: MBZ
	25:16	**Global Outer Loop Stride Y** Format: S9 Vertical stride of the global outer loop, in 2's complement.
	15:10	**Reserved** Format: MBZ
	9:0	**Global Outer Loop Stride X** Format: S9 Horizontal stride of the global outer loop, in 2's complement.
16	31:26	**Reserved** Format: MBZ
	25:16	**Global Inner Loop Unit Y** Format: S9 Vertical stride of the global inner loop, in 2's complement.
	15:10	**Reserved** Format: MBZ
	9:0	**Global Inner Loop Unit X** Format: S9 Horizontal stride of the global inner loop, in 2's complement.
17..n	31:0	**Inline Data**

MEDIA_STATE_FLUSH

MEDIA_STATE_FLUSH

Project:	CHV, BSW
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. It should 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

DWord	Bit	Description
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 MEDIA_STATE_FLUSH Format: OpCode
	23:16	**SubOpcode** Default Value: 4h MEDIA_STATE_FLUSH SubOp Format: OpCode
	15:0	**DWord Length** Project: All Format: =n Total Length - 2 Value / Name / Description: 0h / DWORD_COUNT_n **[Default]** / Excludes DWord (0,1)
1	31:9	**Reserved** Project: All

MEDIA_STATE_FLUSH

		Format: MBZ
	8	**Reserved** Project: N, CHV, BSW
	7	**Flush to GO** Project: CHV, BSW 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** (Project: CHV, BSW*:A0): Use PIPE_CONTROL with CS stall and DC flush bits set, in place of MEDIA_STATE_FLUSH with Flush-to-GO set, to work around a preemption boundary condition.
	6	**Watermark Required** Project: All 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** (Project: CHV, BSW): 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. **Programming Notes** (Project: CHV, BSW*:A0): 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.

MEDIA_VFE_STATE

MEDIA_VFE_STATE

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

A stalling PIPE_CONTROL is required before MEDIA_VFE_STATE unless the only bits that are changed are scoreboard related: Scoreboard Enable, Scoreboard Type, Scoreboard Mask, Scoreboard * Delta. For these scoreboard related states, a MEDIA_STATE_FLUSH is sufficient.

- MEDIA_STATE_FLUSH (optional, only if barrier dependency is needed)
- MEDIA_INTERFACE_DESCRIPTOR_LOAD (optional)

DWord	Bit	Description
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 MEDIA_VFE_STATE Format: OpCode
	23:16	**SubOpcode** Default Value: 0h MEDIA_VFE_STATE SubOp Format: OpCode
	15:0	**DWord Length** Format: =n Total Length - 2 Value: 07h; Name: DWORD_COUNT_n **[Default]**; Description: Excludes DWord (0,1)
1	31:10	**Scratch Space Base Pointer** Format: GeneralStateOffset[31:10] Specifies the 1k-byte aligned address offset to scratch space for use by the kernel. This pointer is relative to the **General State Base Address.**
	9:8	**Reserved** Format: MBZ
	7:4	**Stack Size**

MEDIA_VFE_STATE

DWord	Bit	Description
		Value: [0,11] — Name: — Description: indicating [1KBytes, 2MBytes] **Programming Notes** Since the stack uses the upper portion of the scratch space, **Stack Size =< Per Thread Scratch Space**
	3:0	**Per Thread Scratch Space** Format: U4 Specifies the amount of scratch space allowed to be used by each thread. The driver must allocate enough contiguous scratch space, pointed to by the Scratch Space Pointer, to ensure that the maximum threads in the device each get Per Thread Scratch Space size without exceeding the driver-allocated scratch space. Value: [0,11] — Name: — Description: indicating [1k bytes, 2 Mbytes]: 0 -> 1k, 1->2k, 2->4k, 3->8k ... 11->2M — Project: CHV, BSW
2	31:16	**Reserved** Format: MBZ
	15:0	**Scratch Space Base Pointer High** Format: GeneralStateOffset[47:32] This field specifies the high 16 bits of starting address of the Scratch Space Base Pointer
3	31:16	**Maximum Number of Threads** Format: U16-1 representing thread count Range: [0, n-1] where n = (# EUs) * (# threads/EU). See *Graphics Processing Engine* for listing of #EUs and #threads in each device. Specifies the maximum number of simultaneous root threads allowed to be active. Used to avoid potential deadlock. If child threads are not planning on being used then this field can be set to its maximum value and there will be no thread limit beyond what is currently available in the system; the maximum value can include threads in slices that have been shut down for power reasons. For GPGPU threads the maximum value must be used. **Programming Notes** MSB will be zero due to the range limit below.
	15:8	**Number of URB Entries** Format: U8 Specifies the number of URB entries that are used by the unit. Value: [1,64] — Name: — Description: [1,64] Entries — Project: CHV, BSW

MEDIA_VFE_STATE

DWord	Bit	Description
		Programming Notes Please note that 0 is not allowed for this field.
	7	**Reset Gateway Timer** This field controls the reset of the timestamp counter maintained in Message Gateway. Value | Name 0h | Maintaining the existing timestamp state 1h | Resetting relative timer and latching the global timestamp
	6	**Bypass Gateway Control** Project: CHV, BSW This field configures Gateway to use a simple message protocol. Value | Name 0h | Maintaining OpenGateway/ForwardMsg/CloseGateway protocol (legacy mode) 1h | Bypassing OpenGateway/CloseGateway protocol
	5:3	**Reserved** Project: CHV, BSW Format: MBZ
	2	**Reserved** Format: MBZ
	1:0	**Reserved**
4	31:8	**Reserved**
	7:4	**Reserved** Format: MBZ
	3:2	**Reserved** Project: CHV, BSW Format: MBZ
	1:0	**Slice Disable** This field disables dispatch to slices and subslices for Media and GPGPU applications. It is used to limit the amount of scratch space that needs to be allocated for a context. If a particular configuration doesn't have slice or subslice then there is no impact to disabling it. Value | Name | Description 00b | | All subslices are enabled. 01b | | Slice 2 and 1 are disabled, only Slice 0 with all subslices is enabled. 10b | | Reserved 11b | | Slice 2 and 1 are disabled, only Slice 0 with only subslice 0 enabled.
5	31:16	**URB Entry Allocation Size** Format: U16

MEDIA_VFE_STATE

DWord	Bit	Description
		Specifies the length of each URB entry used by the unit, in 256-bit register increments. ROB address for URB starts after CURBE Allocated region. (URB Entry Allocation Size * Number of URB Entries) + CURBE Allocation Size + Number of Interface Descriptors) must be <= (number of bytes allocated for the URB in L3CNTLREG / 32 bytes per entry). Note: Number of Interface Descriptors is 64. If SLM is enabled for GPGPU work then the number of available entries will be 1/2 the maximum URB entries. **Programming Notes** When Inline data is used with MEDIA_OBJECT or MEDIA_OBJECT_WALKER, then the URB entry allocation size must match the Inline data size. If Indirect data is being used with MEDIA_OBJECT or GPGPU_WALKER then the allocation size must be sufficient for the Indirect data. If both Inline and Indirect are being used, then the allocation size must match the sum of the Inline and Indirect.
	15:0	**CURBE Allocation Size** Format: U16 Specifies the total length allocated for CURBE, in 256-bit register increments. ROB address for CURBE starts at address 64. (URB Entry Allocation Size * Number of URB Entries) + CURBE Allocation Size + Interface Descriptor Entries) must be less than or equal to the number of entries in the URB as described in **Configurations**. Interface Descriptor Entries is 64 If SLM is enabled for GPGPU work then the number of available entries will be ½ the maximum URB entries. **Programming Notes** CURBE Allocation Size should be 0 for GPGPU workloads that uses indirect instead of CURBE.
6	31	**Scoreboard Enable** Project: This field enables and disables the hardware scoreboard in the Media Pipeline. If this field is cleared, hardware ignores the following scoreboard state fields. Value / Name: 0h: Scoreboard disabled 1h: Scoreboard enabled
	30	**Scoreboard Type** Project: This field selects the type of scoreboard in use. Value / Name: 0h: Stalling Scoreboard 1h: Non-Stalling Scoreboard
	29:16	**Reserved**

MEDIA_VFE_STATE		
	15:8	**Reserved** Project: CHV, BSW Format: MBZ
	7:0	**Scoreboard Mask** Project: Format: Enable[8] Each bit indicates the corresponding dependency scoreboard is enabled. The scoreboard is based on the relative (X, Y) distance from the current threads' (X, Y) position. **Bit n (for n = 0...7):** Score n is enabled.
7	31:28	**Scoreboard 3 Delta Y** Project: Format: S3 Relative vertical distance of the dependent instance assigned to scoreboard 3, in the form of 2's compliment. **Programming Notes** MBZ if scoreboard is disabled.
	27:24	**Scoreboard 3 Delta X** Project: Format: S3 Relative horizontal distance of the dependent instance assigned to scoreboard 3, in the form of 2's compliment. **Programming Notes** MBZ if scoreboard is disabled.
	23:20	**Scoreboard 2 Delta Y** Project: Format: S3 Relative vertical distance of the dependent instance assigned to scoreboard 2, in the form of 2's compliment. **Programming Notes** MBZ if scoreboard is disabled.
	19:16	**Scoreboard 2 Delta X** Project: Format: S3 Relative horizontal distance of the dependent instance assigned to scoreboard 2, in the form of 2's compliment. **Programming Notes** MBZ if scoreboard is disabled.

MEDIA_VFE_STATE		
	15:12	**Scoreboard 1 Delta Y** Project: | Format: | S3 Relative vertical distance of the dependent instance assigned to scoreboard 1, in the form of 2's compliment. **Programming Notes** MBZ if scoreboard is disabled.
	11:8	**Scoreboard 1 Delta X** Project: | Format: | S3 Relative horizontal distance of the dependent instance assigned to scoreboard 1, in the form of 2's compliment. **Programming Notes** MBZ if scoreboard is disabled.
	7:4	**Scoreboard 0 Delta Y** Project: | Format: | S3 Relative vertical distance of the dependent instance assigned to scoreboard 0, in the form of 2's compliment. **Programming Notes** MBZ if scoreboard is disabled.
	3:0	**Scoreboard 0 Delta X** Project: | Format: | S3 Relative horizontal distance of the dependent instance assigned to scoreboard 0, in the form of 2's compliment. **Programming Notes** MBZ if scoreboard is disabled.
8	31:28	**Scoreboard 7 Delta Y** Project: | Format: | S3 Relative vertical distance of the dependent instance assigned to scoreboard 7, in the form of 2's compliment. **Programming Notes** MBZ if scoreboard is disabled.
	27:24	**Scoreboard 7 Delta X** Project: | Format: | S3

MEDIA_VFE_STATE

		Relative horizontal distance of the dependent instance assigned to scoreboard 7, in the form of 2's compliment. **Programming Notes** MBZ if scoreboard is disabled.
	23:20	**Scoreboard 6 Delta Y** Project: Format: S3 Relative vertical distance of the dependent instance assigned to scoreboard 6, in the form of 2's compliment. **Programming Notes** MBZ if scoreboard is disabled.
	19:16	**Scoreboard 6 Delta X** Project: Format: S3 Relative horizontal distance of the dependent instance assigned to scoreboard 6, in the form of 2's compliment. **Programming Notes** MBZ if scoreboard is disabled.
	15:12	**Scoreboard 5 Delta Y** Project: Format: S3 Relative vertical distance of the dependent instance assigned to scoreboard 5, in the form of 2's compliment. **Programming Notes** MBZ if scoreboard is disabled.
	11:8	**Scoreboard 5 Delta X** Project: Format: S3 Relative horizontal distance of the dependent instance assigned to scoreboard 5, in the form of 2's compliment. **Programming Notes** MBZ if scoreboard is disabled.
	7:4	**Scoreboard 4 Delta Y** Project: Format: S3 Relative vertical distance of the dependent instance assigned to scoreboard 4, in the form of 2's compliment.

MEDIA_VFE_STATE

		Programming Notes MBZ if scoreboard is disabled.
	3:0	**Scoreboard 4 Delta X** Project: Format: S3 Relative horizontal distance of the dependent instance assigned to scoreboard 4, in the form of 2's compliment. **Programming Notes** MBZ if scoreboard is disabled.

Media Block Read MSD

MSD1R_MB - Media Block Read MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Other
Group:	Media Block R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 04h Project: All Format: Opcode Media Block Read message
	13:11	**Reserved** Project: All Format: MBZ Ignored
	10:8	**Vertical Line Stride Override** Project: All Format: MDC_VLSO [CHV, BSW] If enabled, specifies the Vertical Line Stride and Vertical Line Stride Offset override fields.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Media Block Write MSD

MSD1W_MB - Media Block Write MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Other
Group:	Media Block R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 0Ah Project: All Format: Opcode Media Block Write message
	13:11	**Reserved** Project: All Format: MBZ Ignored
	10:8	**Vertical Line Stride Override** Project: All Format: MDC_VLSO [CHV, BSW] If enabled, specifies the Vertical Line Stride and Vertical Line Stride Offset override fields.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Media Transpose Read MSD

MSD1R_TT - Media Transpose Read MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Other
Group:	Transpose Read

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 00h Project: All Format: Opcode Transpose Read message
	13:8	**Reserved** Project: All Format: MBZ Ignored
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Memory Fence MSD

MSD_MEMFENCE - Memory Fence MSD

Project:	CHV, BSW
Source:	DataPort 0
Length Bias:	1
Family:	Other
Group:	Memory Fence

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] Indicates that the message requires a header.
	18	**Legacy Message** Default Value: 0h Project: All Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 07h Project: All Format: Opcode Memory Fence message
	13	**Commit** Project: All Format: Enable Specifies whether control is returned to the thread only after the fence has been honored.
	12:9	**L3 Flush** The L3 Flush control is one of the following GSYNC signals. Value: 0h; Name: Disabled **[Default]**; Description: The L3 caches are not flushed. **Programming Notes** If multiple caches need to be flushed, the commands need to be sent separately.
	8	**Reserved**
	7:0	**Reserved**

L3 Flush values:

Value	Name	Description
0h	Disabled **[Default]**	The L3 caches are not flushed.

Programming Notes
If multiple caches need to be flushed, the commands need to be sent separately.

MFC_AVC_PAK_OBJECT

MFC_AVC_PAK_OBJECT

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFC_AVC_PAK_OBJECT Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h AVC_ENC Format: OpCode
	23:21	**SubOpcode A** Default Value: 2h Format: OpCode
	20:16	**SubOpcode B** Default Value: 9h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Length -2 Value: 000Ah — Name: DWORD_COUNT_n **[Default]** — Project: CHV, BSW

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

MFC_JPEG_ HUFF_TABLE_STATE

MFC_JPEG_ HUFF_TABLE_STATE		
Project:	CHV, BSW	
Source:	VideoCS	
Length Bias:	2	

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFC_JPEG_ HUFF_TABLE_STATE Format: OpCode
	26:24	**Media Command Opcode** Default Value: 7h JPEG Format: OpCode
	23:21	**SubOpcode A** Default Value: 2h Common Format: OpCode
	20:16	**SubOpcode B** Default Value: 3h MEDIA_ Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 0AEh Excludes DWord (0,1) Format: =n Total Length - 2
1	31:1	**Reserved** Format: MBZ
	0	**Huff Table ID** Format: U1 Huffman table destination identifier will specify one of two destinations at the encoder into which the Huffman table must be stored.

<table>
<tr><th colspan="3">MFC_JPEG_ HUFF_TABLE_STATE</th></tr>
<tr><td></td><td></td><td><table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td></td><td>Huffman table 0</td></tr><tr><td>1</td><td></td><td>Huffman table 1</td></tr></table></td></tr>
<tr><td>2..13</td><td>31:0</td><td>DC_TABLE<table><tr><td>Format:</td><td>3Bytes: Byte0 for Code length, Byte1 and Byte2 for Code word, and Byte3 for dummy</td></tr></table>12 categories with code length and code word. Each run/size has 1-byte code length, and 2-byte code word.</td></tr>
<tr><td>14..175</td><td>31:0</td><td>AC_TABLE<table><tr><td>Format:</td><td>3Bytes: Byte0 for Code length, Byte1 and Byte2 for Code word, and Byte3 for dummy</td></tr></table>162 run/size with code length and code word. Each run/size has 1-byte code length, and 2-byte code word.</td></tr>
</table>

MFC_JPEG_SCAN_OBJECT

MFC_JPEG_SCAN_OBJECT

Project: CHV, BSW
Source: VideoCS
Length Bias: 2

Encoder Only

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFC_JPEG_SCAN_OBJECT Format: OpCode
	26:24	**Media Command Opcode** Default Value: 7h JPEG_ENC Format: OpCode
	23:21	**SubOpcode A** Default Value: 2h Format: OpCode
	20:16	**SubOpcode B** Default Value: 9h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 001h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:26	**Reserved** Format: MBZ
	25:0	**MCU Count** Format: U26 This field indicates the number of MCUs in the Scan. MCU Count = M_x X M_y The number of MCUs in a row: $M_x = (X + (H_1 *8 -1)) / (H_1 *8)$ The number of MCUs in a column: $M_y = (Y + (V_1 *8 - 1)) / (V_1 *8)$*X*: The number of samples per line in Y-image *Y*: The number of lines in Y-image *H1*: Horizontal sampling factor of Y-image in the Frame header *V1*: Vertical sampling factor of Y-image in the Frame header

<table>
<tr><th colspan="3">MFC_JPEG_SCAN_OBJECT</th></tr>
<tr><td rowspan="4">2</td><td>31:25</td><td>Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>24:22</td><td>Huffman AC Table
<table>
<tr><td>AC Huffman table destination selector specifies one of two possible AC table destinations for each Y, U, V, or R, G, B. The AC Huffman tables must have been loaded in destination 0 and 1 by the time of issuing MFC_JPEG_HUFF_TABLE_STATE Command.</td></tr>
<tr><td>If AC table 0 is used for Y and AC table 1 is used for U and V, it will be set to 110b. If AC table 0 is used for R, G, and B, it will be set to 000b and so on. Refer to the table below for the summary of actions.</td></tr>
</table>
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0XXb</td><td>Bit24 (V0)</td><td>The third image component must use the AC table 0.</td></tr>
<tr><td>1XXb</td><td>Bit24 (V1)</td><td>The third image component must use the AC table 1.</td></tr>
<tr><td>X0Xb</td><td>Bit23 (U0)</td><td>The second image component must use the AC table 0.</td></tr>
<tr><td>X1Xb</td><td>Bit23 (U1)</td><td>The second image component must use the AC table 1.</td></tr>
<tr><td>XX0b</td><td>Bit22 (Y0)</td><td>The first image component must use the AC table 0.</td></tr>
<tr><td>XX1b</td><td>Bit22 (Y1)</td><td>The first image component must use the AC table 1.</td></tr>
</table>
<table>
<tr><th>Restriction</th></tr>
<tr><td>When InputSurfaceFormatYUV = RGB, because the order of input image components can be RGB, GBR, BGR,\ or YUV, Bit22 is used for the first image component, Bit23 is used for the second image component, and Bit24 is used for the third image component.</td></tr>
</table></td></tr>
<tr><td>21</td><td>Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>20:18</td><td>Huffman DC Table
<table>
<tr><td>DC Huffman table destination selector specifies one of two possible DC table destinations for each Y, U, V, or R, G, B. The DC Huffman tables shall have been loaded in destination 0 and 1 by the time of issuing MFC_JPEG_HUFF_TABLE_STATE Command.</td></tr>
<tr><td>if DC table 0 is used for Y and DC table 1 is used for U and V, it will be set to 110b. If DC table 0 is used for R, G, and B, it will be set to 000b and so on. Refer to the table below for the summary of actions.</td></tr>
</table>
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0XXb</td><td>Bit20 (V0)</td><td>The third image component must use the DC table 0.</td></tr>
<tr><td>1XXb</td><td>Bit20 (V1)</td><td>The third image component must use the DC table 1.</td></tr>
<tr><td>X0Xb</td><td>Bit19 (U0)</td><td>The second image component must use the DC table 0.</td></tr>
<tr><td>X1Xb</td><td>Bit19 (U1)</td><td>The second image component must use the DC table 1.</td></tr>
<tr><td>XX0b</td><td>Bit18 (Y0)</td><td>The first image component must use the DC table 0.</td></tr>
<tr><td>XX1b</td><td>Bit18 (Y1)</td><td>The first image component must use the DC table 1.</td></tr>
</table></td></tr>
</table>

MFC_JPEG_SCAN_OBJECT

Restriction
When InputSurfaceFormatYUV = RGB, because the order of input image components can be RGB, GBR, BGR, YUV, **Bit18** is used for the first image component, **Bit19** is used for the second image component, and **Bit20** is used for the third image component.

17 **Head Present Flag**

If this flag is set to 0, then no MFC_JPEG_PAK_INSERT_OBJECT commands will be sent. If this flag is set to 1, then one or more MFC_JPEG_PAK_INSERT_OBJECT commands will be sent after MFC_JPEG_SCAN_OBJECT command.

Value	Name	Description
0		No insertion into the output bitstream buffer before Scan encoded bitstream
1		Headers, tables, App data insertion into the output bitstream buffer. HW will insert the insertion data before the Scan encoded bitstream.

16 **Is Last Scan**

If this flag is set, then HW will insert EOI (0xFFD9) to the end of Scan encoded bitstream.

Value	Name	Description
0		Not the last Scan.
1		Indicates that the current Scan is the last one.

15:0 **Restart Interval**

Format:	U16

Specifies the number of MCUs in an ECS, except for the last ECS. Restart maker is inserted periodically and it separates the two neighboring ECSs.

Value	Name
0-FFFFh	

Programming Notes
A value of '0' implies that the Scan Data has a single ECS.

MFC_MPEG2_PAK_OBJECT

MFC_MPEG2_PAK_OBJECT

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

The MFC_MPEG2_PAK_OBJECT command is the second primitive command for the MPEG-2 Encoding Pipeline. Different from AVC, the MV Data portion of the bitstream is loaded as part of MB control data.
Before issuing a MFC_MPEG2_PAK_OBJECT command, all MPEG2_MFX states need to be valid. Therefore the commands used to set these states need to have been issued prior to the issue of this command.
MB record must be consecutive with no gaps, hence we do not need MB(x,y) in each MB command. Internal counter will keep track of the current MB address, starting from the Start_MB_In_Slice loaded at the beginning of each slice.
MFC_ MPEG2_PAK_OBJECT command follows the MbType definition like MFD. Many fields in this command are identical to that in VME output. This is intended to reduce software converting overhead from VME to PAK.
Encoding statistical data such as the total size of the output bitstream are provided through MMIO registers. Software may access these registers through MI_STORE_REGISTER_MEM command.

DWord	Bit	Description	
0	31:29	**Command Type**	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	**Pipeline**	
		Default Value:	2h MFC_AVC_PAK_INSERT_OBJECT
		Format:	OpCode
	26:24	**Media Command Opcode**	
		Default Value:	3h MPEG2
		Format:	OpCode
	23:21	**SubOpcode A**	
		Default Value:	2h ENC
		Format:	OpCode
	20:16	**SubOpcode B**	
		Default Value:	9h MEDIA_
		Format:	OpCode
	15:12	**Reserved**	
	11:0	**DWord Length**	
		Default Value:	0007h Excludes DWord (0,1)
		Project:	All
		Format:	=n Total Length - 2

MFC_MPEG2_PAK_OBJECT		
1..8	31:0	**Inline Data** All the required MB level controls and parameters for encoding are captured as inline data of the MFC_MPEG2_PAK_OBJECT command. It has a fixed size of 8 DWs. Its definition is described in the next section

MFC_MPEG2_SLICEGROUP_STATE

MFC_MPEG2_SLICEGROUP_STATE

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

This is a slice group level command and can be issued multiple times within a picture that is comprised of multiple slice groups. The same command is used for AVC encoder (PAK mode) and decoder (VLD and IT modes).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_MPEG2_SLICEGROUP_STATE Format: OpCode
	26:24	**Media Command Opcode** Default Value: 3h MPEG2 Format: OpCode
	23:21	**SubOpcode A** Default Value: 2h MEDIA_ Format: OpCode
	20:16	**SubOpcode B** Default Value: 3h MEDIA_ Format: OpCode
	15:12	**Reserved** Project: All Format: MBZ
	11:0	**DWord Length** Default Value: 6h Excludes DWord (0,1) Project: All Format: =n Total Length - 2
1	31	**MbRateCtrlFlag- RateControlCounterEnable (Encoder-only)** To enable the accumulation of bit allocation for rate controlThis 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.Note: To reset MB level rate control (QRC), we need to set both bits MbRateCtrlFlag and MbRateCtrlReset to 1 in the new slice

MFC_MPEG2_SLICEGROUP_STATE

Value	Name	Project
0h	Disable	All
1h	Enable	All

30 — **MbRateCtrlReset- ResetRateControlCounter (Encoder-only)**

To reset the bit allocation accumulation counter to 0 to restart the rate control.

Value	Name	Description	Project
0h	Disable	Not reset	All
1h	Enable	reset	All

29:28 — **MbRateCtrlMode- RC Triggle Mode (Encoder-only)**

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 — **MbRateCtrlParam- RC Stable Tolerance (Encoder-only)**

Format:	U4

This field specifies the tolerance required to deactivate RC once it has been triggered.

Value	Name
[0, 15]	

23 — **RateCtrlPanicFlag - RC Panic Enable (Encoder-only)**

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	Project
0	Disable	All
1	Enable	All

22 — **RateCtrlPanicType - RC Panic Type (Encoder-only)**

This field selects between two RC Panic methods. 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.

Value	Name	Description
0h		QP Panic
1h		CBP Panic

21 — **Reserved**

<table>
<tr><th colspan="3">MFC_MPEG2_SLICEGROUP_STATE</th></tr>
<tr><td></td><td>20</td><td>SkipConvDisabled - MB Type Skip Conversion Disable (Encoder-only)
This field is only valid for a P or B slice. It must be zero for other slice types. Rules are provided in Section 2.3.3.1.6
<table>
<tr><th>Value</th><th>Name</th><th>Description</th><th>Project</th></tr>
<tr><td>0h</td><td>Enable</td><td>Enable skip type conversion</td><td>All</td></tr>
<tr><td>1h</td><td>Disable</td><td>Disable skip type conversion</td><td>All</td></tr>
</table></td></tr>
<tr><td></td><td>19</td><td>IsLastSliceGrp
IsLastSliceGrp = 1 if the current slice group is the last slice group of a picture; 0 otherwise.It is used by the zero filling in the Minimum Frame Size test.</td></tr>
<tr><td></td><td>18</td><td>BitstreamOutputFlag - Compressed BitStream Output Disable Flag (Encoder-only)
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Enable</td><td>enable the writing of the output compressed bitstream</td></tr>
<tr><td>1h</td><td>Disable</td><td>disable the writing of the output compressed bitstream</td></tr>
</table></td></tr>
<tr><td></td><td>17</td><td>HeaderPresentFlag - Header Insertion Present in Bitstream (Encoder-only)
<table>
<tr><th>Value</th><th>Name</th><th>Description</th><th>Project</th></tr>
<tr><td>0h</td><td>Disable</td><td>no header insertion into the output bitstream buffer, in front of the current slice encoded bits</td><td>All</td></tr>
<tr><td>1h</td><td>Enable</td><td>header insertion into the output bitstream buffer is present, and is in front of the current slice encoded bits.</td><td>All</td></tr>
</table></td></tr>
<tr><td></td><td>16</td><td>SliceData PresentFlag - SliceData Insertion Present in Bitstream (Encoder-only)
<table>
<tr><th>Value</th><th>Name</th><th>Description</th><th>Project</th></tr>
<tr><td>0h</td><td>Disable</td><td>no Slice Data insertion into the output bitstream buffer</td><td>All</td></tr>
<tr><td>1h</td><td>Enable</td><td>Slice Data insertion into the output bitstream buffer is present.</td><td>All</td></tr>
</table></td></tr>
<tr><td></td><td>15</td><td>TailPresentFlag - Tail Insertion Present in bitstream (Encoder-only)
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td></td><td>no tail insertion into the output bitstream buffer, after the current slice encoded bits</td></tr>
<tr><td>1h</td><td></td><td>tail insertion into the output bitstream buffer is present, and is after the current slice encoded bits.</td></tr>
</table></td></tr>
<tr><td></td><td>14</td><td>FirstSliceHdrDisabled
when this is on, the first slice header of the slice group is expected to be provided by the user via insertion command. PAK HW will skip it.</td></tr>
<tr><td></td><td>13</td><td>IntraSlice
intra slice value included in slice headers, when IntraSliceFlag = 1.</td></tr>
<tr><td></td><td>12</td><td>IntraSliceFlag
intra slice flag included in slice headers</td></tr>
<tr><td></td><td>11:8</td><td>Reserved
<table>
<tr><td>Format:</td><td>MBZ for SliceID extension</td></tr>
</table></td></tr>
</table>

MFC_MPEG2_SLICEGROUP_STATE

	7:4	**SliceID[3:0] (Encoder-only)** To identify the output data (coding information record) returned for rate control from PAK to ENC and VPP
	3:2	**Reserved** Format: MBZ for StreamID extension
	1:0	**StreamID[1:0] (Encoder-only)** To identify the output data (coding information record) returned for rate control from PAK to ENC and VPP
2	31:24	**NextSgMbYcnt - also NextStartVertPos** Vertical count of the first MB in the next slice group (Encoder-only)Note: This field restricts total number of MB in the Y direction to 255 or less.
	23:16	**NextSgMbXcnt - also NextStartHorzPos** BitFieldDesc
	15:8	**FirstMbYcnt - also CurrStartVertPos** Project: All Format: U8 also CurrStartVertPos, Vertical count of the first MB in the current slice group (Encoder-only)
	7:0	**FirstMbXcnt - also CurrStartHorzPos** Project: All Format: U8 Horizontal count of the first MB in the current slice group (Encoder-only)
3	31:9	**Reserved** Format: MBZ
	8	**SliceGroupSkip** Project: All Exists If: //Encoder Only Format: U1 All macroblocks are skipped
	7:6	**Reserved** Format: MBZ
	5:0	**SliceGroupQp** Project: All Exists If: //Encoder Only Format: U6 Initial slice quality parameter

MFC_MPEG2_SLICEGROUP_STATE		
4	31:29	**Reserved** Format: MBZ
	28:0	**BitstreamOffset - Indirect PAK-BSE Data Start Address (Write)** Exists If: //Encoder Only 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.This field is only valid for AVC encode mode. Value: [0,512MB] — Name:
5	31:24	**MaxQpNegModifier - Magnitude of QP Max Negative Modifier (Encoder-only)** Format: U8 This field specifies the lower limit of the QP modifier. Value: [0, 51] — Name:
	23:16	**MaxQpPosModifier - Magnitude of QP Max Positive Modifier (Encoder-only)** Format: U8 This field specifies the upper limit of the QP modifier. Value: [0, 51] — Name:
	15:12	**ShrinkParam - Shrink Resistance (Encoder-only)** Format: U4 This field specifies the additional points added each time decreased correction is invoked. Value: [0, 15] — Name:
	11:8	**Shrinkaram - Shrink Init (Encoder-only)** Format: U4 This field specifies the initial points required to trip decreased control. Value: [0, 15] — Name:
	7:4	**GrowParam - Grow Resistance (Encoder-only)** Format: U4 This field specifies the additional points added each time increased correction is invoked. Value: [0, 15] — Name:

MFC_MPEG2_SLICEGROUP_STATE

DWord	Bit	Description
	3:0	**GrowParam - Grow Init (Encoder-only)** Format: U4 This field specifies the initial points required to trip increased control. Value: [0, 15] — Name:
6	31:24	**Reserved** Format: MBZ
	23:20	**CorrectPoints - Correct 6 (Encoder-only)** Format: U4 This field specifies the points used in the lowermost RC region when sum_act <= sum_min. Value: [0, 15] — Name:
	19:16	**CorrectPoints - Correct 5 (Encoder-only)** Format: U4 This field specifies the points used in the fifth RC region when sum_act > sum_min but <= lower_midpt. Value: [0, 15] — Name:
	15:12	**CorrectPoints - Correct 4 (Encoder-only)** Format: U4 This field specifies the points used in the fourth RC region when sum_act > lower_midpt but <= sum_target. Value: [0, 15] — Name:
	11:8	**CorrectPoints - Correct 3 (Encoder-only)** Format: U4 This field specifies the points used in the third RC region when sum_act > sum_target but <= upper_midpt. Value: [0, 15] — Name:
	7:4	**CorrectPoints - Correct 2 (Encoder-only)** Format: U4 This field specifies the points used in the second RC region when sum_act > upper_midpt but <= sum_max. Value: [0, 15] — Name:
	3:0	**CorrectPoints - Correct 1 (Encoder-only)** Format: U4

MFC_MPEG2_SLICEGROUP_STATE

DWord	Bit	Description	
		This field specifies the points used in the topmost RC region when sum_act > sum_max	
		Value	**Name**
		[0, 15]	
7	31:28	**CV7 - Clamp Value 7 (Encoder-only)**	
		Exists If:	//Encoder Only
	27:24	**CV6 - Clamp Value 6 (Encoder-only)**	
		Project:	All
		Exists If:	//Encoder Only
		Format:	U4
	23:20	**CV5 - Clamp Value 5 (Encoder-only)**	
		Project:	All
		Exists If:	//Encoder Only
		Format:	U4
	19:16	**CV4 - Clamp Value 4 (Encoder-only)**	
		Project:	All
		Exists If:	//Encoder Only
		Format:	U4
	15:12	**CV3 - Clamp Value 3 (Encoder-only)**	
		Project:	All
		Exists If:	//Encoder Only
		Format:	U4
	11:8	**CV2 - Clamp Value 2 (Encoder-only)**	
		Project:	All
		Exists If:	//Encoder Only
		Format:	U4
	7:4	**CV1 - Clamp Value 1 (Encoder-only)**	
		Project:	All
		Exists If:	//Encoder Only
		Format:	U4
	3:0	**CV0 - Clamp Value 0 (Encoder-only)** If the magnitude of coefficients at locations assigned with CV0 (mapping shown below) exceeds 2CV0-1, they are replaced with 2CV0-1. 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 8x8 frame block, each coefficient is mapped to one of the eight CV values as following:	

none	none	CV7	CV6	CV5	CV4	CV3	CV3

MFC_MPEG2_SLICEGROUP_STATE

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

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

MFD_AVC_BSD_OBJECT

MFD_AVC_BSD_OBJECT

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

Description
The MFD_AVC_BSD_OBJECT command is the only primitive command for the AVC Decoding Pipeline. The same command is used for both CABAC and CAVLD modes. The Slice Data portion of the bitstream is loaded as indirect data object.Before issuing a MFD_AVC_BSD_OBJECT command, all AVC states of the MFD Engine need to be valid. Therefore the commands used to set these states need to have been issued prior to the issue of a MFD_AVC_BSD_OBJECT command.
Context switch interrupt is not supported by this command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFD_AVC_BSD_OBJECT Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h AVC_DEC Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h Format: OpCode
	20:16	**SubOpcode B** Default Value: 8h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Total Length - 2 Value: 4h — Name: Excludes DWord (0,1) = 0004 **[Default]** — Project: CHV, BSW

MFD_AVC_BSD_OBJECT

DWord	Bit	Description
1	31:0	**Indirect BSD Data Length** Format: U32 This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. This field must have the same alignment as the Indirect Object Data Start Address. AVC Short Format : It is the length in bytes of the bitstream data for the current slice, including Slice Header + Slice Data + Emulation Prevention Bytes + any filling trailing zeros after the last MB. Hardware ignores the contents after the last non-zero byte. Trailing zero is allowed and handled correctly in both CABAC and CAVLC modes.
2	31:29	**Reserved** Format: MBZ
	28:0	**Indirect BSD Data Start Address** Project: CHV, BSW Format: U29 This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This pointer is relative to the **MFD Indirect Object Base Address**. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the AVC bitstream data in both CABAC/CAVLD Modes. In implementing a phantom slice at the end of a picture for automatic error concealment, this field should set to 0. It includes the NAL Header (the NAL Header does not need to perform EMU detection). For AVC and SVC Base Layer, it is a single byte. But for SVC and MVC, the NAL Header is 4 Bytes long. These NAL Header Unit must be passed to HW in the compressed bitstream buffer. Value: [0,512MB) — Name: (blank)
3..5	31:0	**Inline Data** All the required Slice Header parameters and error handling settings are captured as InLine Data of the AVC_BSD_OBJECT command. It has a fixed size of 4 DWs. Its definition is described in the follwoing section: Inline Data Description [CHV, BSW].
6		

MFD_AVC_DPB_STATE

MFD_AVC_DPB_STATE		
Project:	CHV, BSW	
Source:	VideoCS	
Length Bias:	2	

This is a frame level state command used only in DXVA2 AVC Short Slice Bitstream Format VLD mode. RefFrameList[16] of DXVA2 interface is replaced with intel Reference Picture Addresses[16] of MFX_PIPE_BUF_ADDR_STATE command. The LongTerm Picture flag indicator of all reference pictures are collected into LongTermPic_Flag[16]. FieldOrderCntList[16][2] and CurrFieldOrderCnt[2] of DXVA2 interface are replaced with intel POCList[34] of MFX_AVC_DIRECTMODE_STATE command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_MULTI_DW Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h AVC_DEC Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h Format: OpCode
	20:16	**SubOpcode B** Default Value: 6h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Total Length - 2

Value	Name	Project
9h	Excludes DWord (0,1) **[Default]**	CHV, BSW

MFD_AVC_DPB_STATE

DWord	Bits	Description
1	31:16	**LongTermFrame_Flag[16][1 bit]** One-to-one correspondence with the entries of the Intel RefFrameList[16]. 1 bit per reference frame.

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

DWord	Bits	Description
1	15:0	**Non-ExistingFrame_Flag[16][1 bit]** One-to-one correspondence with the entries of the Intel RefFrameList[16]. 1 bit per reference frame.

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

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

DWord	Bits	Description
2	31:0	**UsedForReference_Flag[16][2 bits]** One-to-one correspondence with the entries of the Intel RefFrameList[16]. 2 bits per reference frame.

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

DWord	Bits	Description
3..10	31:0	**LTSTFrameNumList[16][16 bits]** One-to-one correspondence with the entries of the Intel RefFrameList[16]. 16 bits per reference frame.Depending on the corresponding LongTermFrame_Flag[], the content of this field is interpreted differently.

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

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

MFD_AVC_DPB_STATE		
11..18 **Project:** CHV, BSW	31:0	**ViewIDList[16][16 bits]** Project: CHV, BSW One-to-one correspondence with the entries of the Intel RefFrameList[16]. 16 bits per reference frame. The view ids are 10-bits, the upper 6 bits are ignored."000000" & ViewId1[9:0] & "000000" & ViewId0[9:0] **Programming Notes** When an Intel RefFrameList[i] is not an valid entries, Viewid should be set to 0x00
19..22 **Project:** CHV, BSW	31:0	**ViewOrderListL0[16][8 bits]** Project: CHV, BSW One-to-one correspondence with the entries of the Intel RefFrameList[16]. 8 bits per reference frame. The view order need 4-bits, the upper 4 bits are ignored. 0000 & ViewOrder3[3:0] & 0000 & ViewOrder2[3:0] & 0000 & ViewOrder1[3:0] & 0000 & ViewOrder0[3:0] **Programming Notes** When the ViewOrderListL0[i] is not an valid inter-view reference, its corresponding ViewOrder should be set to 0xF Since only interview with the same polarity will be used, there is no need to have field bit in this list. Hardware is going to append correct polarity bit as needed.
23..26 **Project:** CHV, BSW	31:0	**ViewOrderListL1[16][8 bits]** Project: CHV, BSW One-to-one correspondence with the entries of the Intel RefFrameList[16]. 8 bits per reference frame. The view order need 4-bits, the upper 4 bits are ignored. 0000 & ViewOrder3[3:0] & 0000 & ViewOrder2[3:0] & 0000 & ViewOrder1[3:0] & 0000 & ViewOrder0[3:0] **Programming Notes** When the ViewOrderListL1[i] is not an valid inter-view reference, its corresponding ViewOrder should be set to 0xF Since only interview with the same polarity will be used, there is no need to have field bit in this list. Hardware is going to append correct polarity bit as needed.

MFD_AVC_PICID_STATE

MFD_AVC_PICID_STATE

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

This is a frame level state command used for both AVC Long and Short Format in VLD mode.PictureID[16] contains the pictureID of each reference picture (16 maximum) so hardware can uniquely identify the reference picture across frames (this will be used for DMV operation).This command will be needed for both short and long format.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_MULTI_DW Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h MFD_AVC_DPB_STATE Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h DEC Format: OpCode
	20:16	**SubOpcode B** Default Value: 5h MEDIA_ Format: OpCode
	15:12	**Reserved** Project: All Format: MBZ
	11:0	**DWord Length** Default Value: 0008h Excludes DWord (0,1) Project: CHV, BSW Format: =n Total Length - 2
1	31:1	**Reserved** Project: All Format: MBZ

MFD_AVC_PICID_STATE		
	0	**PictureID Remapping Disable** Project: CHV, BSW Value: 0h — Name: AVC decoder will use 16 bits Picture ID to handle DMV and identify the reference picture — Description: Desc — Project: All Value: 1h — Name: AVC decoder will use 4 bits FrameStoreID (index to RefFrameList) to handle DMV and identify the reference picture — Description: Desc — Project: All **Programming Notes** If Picture ID Remapping Disable is "1", PictureIDList will not be used.
2..9	31:0	**PictureIDList[16][16 bits]** Project: CHV, BSW One-to-one correspondence with the entries of the Intel RefFrameList[16]. 16 bits per reference frame. PictureID of each Frame uniquely identifies the reference picture across frames. The same number cannot be reused until the referece picture is completely retired(no longer used for referece)When an element of the list of frames is not relevant (e.g., due to the corresponding reference entry being empty or being marked as "not used for reference"), the value of the LTSTFrameNumList entry shall be set to 0.

MFD_AVC_SLICEADDR

MFD_AVC_SLICEADDR	
Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFD_AVC_ SLICEADDR Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h AVC_DEC Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h Format: OpCode
	20:16	**SubOpcode B** Default Value: 7h Format: OpCode
	15:12	**Reserved** Format: MBZ

MFD_AVC_SLICEADDR

DWord	Bit	Description
	11:0	**DWord Length** Format: =n Total Length - 2 Value: 1h — Name: Excludes DWord (0,1) **[Default]** — Project: CHV, BSW
1	31:0	**Indirect BSD Data Length** Project: CHV, BSW Format: U32 This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. Driver always programs this up to 5 bytes; for bitstream less than 5 bytes, driver program the lesser value. (Emulation Prevention Byte should never happen for the first 5 bytes when the max picture size can only be 4Kx4K)It is the length in bytes of the bitstream data for the current slice, including Slice Header + Slice Data + Emulation Prevention Bytes + any filling trailing zeros after the last MB. Hardware ignores the contents after the last non-zero byte. Trailing zero is allowed and handled correctly in both CABAC and CAVLC modes.
2	31:29	**Reserved** Format: MBZ
	28:0	**Indirect BSD Data Start Address** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This pointer is relative to the MFD Indirect Object Base Address.Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the AVC bitstream data in both CABAC/CAVLD Modes.In implementing a phantom slice at the end of a picture for automatic error concealment, this field should set to 0.It includes the NAL Header Byte. (but does not perform EMU detection).Must provide a valid MB address, even if error. MB must be clamped to within a pic boundary. Value: [0,512MB) — Name:

MFD_IT_OBJECT

MFD_IT_OBJECT

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

All weight mode (default and implicit) are mapped to explicit mode. But the weights come in either as explicit or implicit.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFD_IT_OBJECT Format: OpCode
	26:24	**Media Command Opcode** Default Value: 0h MFX_COMMON_DEC Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h Format: OpCode
	20:16	**SubOpcode B** Default Value: 9h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 06h Excludes DWord (0,1) For AVC = Ch Format: =n Total Length - 2 Note: Regardless of the mode, inline data must be present in this command.
1	31:10	**Reserved** Format: MBZ
	9:0	**Indirect IT-MV Data Length** Format: U10 FormatDesc: In bytes This field provides the length in bytes of the indirect data, which contains all the MVs for the current MB (in any partitioning and subpartitioning form). A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect IT-MV Data Start Address field is ignored.

MFD_IT_OBJECT

DWord	Bit	Description
		This field must have the same alignment as the Indirect Object Data Start Address.AVC-IT Mode: It must be DWord aligned (since each MV is 4bytes in size)Driver has to derived this field from MVsize (MVquantity in DXVA, exact size) *4 bytes per MV.This field is only valid in AVC decoder IT mode (VC1 and MPEG uses inline MV data).
2	31:29	**Reserved** Format: MBZ
	28:0	**Indirect IT-MV Data Start Address Offset** This field specifies the memory starting address (offset) of the MV data to be fetched into the IT pipeline for processing. This pointer is relative to the Indirect IT-MV Object Base Address.Hardware ignores this field if indirect data is not present, i.e. the Indirect MV Data Length is set to 0. Alignment of this address depends on the mode of operation.AVC-IT Mode: It must be DWord aligned (since each MV is 4 bytes in size). This field is only valid in AVC decoder IT mode (VC1 and MPEG uses inline MV data). Value: [0,512MB) — Name: (none)
3	31:12	**Reserved** Format: MBZ
	11:0	**Indirect IT-COEFF Data Length** Project: All This field provides the length in bytes of the indirect data, which contains all the non-zero coefficients for the current MB. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect IT-COEFF Data Start Address field is ignored. Since each IT-COEFF data is 1 DW in size, with 12 bits, this field can be extended to support up to 4:4:4 format.(256 pixel * 3 byte pixel components * 4 bytes per coeff).This field must be integer multiple of 16-bytes for AVC (since each coefficient is 4 bytes in size).This field is only valid in AVC, VC1, MPEG2 decoder IT mode. Value: [0,3072] — Name: In bytes [0, 256*3*4]
4	31:29	**Reserved**
	28:0	**Indirect IT-COEFF Data Start Address Offset** Project: All This field specifies the memory starting address (offset) of the coeff data to be loaded into the IT pipeline for processing. This pointer is relative to the Indirect IT-COEFF Object Base Address.Hardware ignores this field if indirect IT-COEFF data is not present, i.e. the Indirect IT-COEFF Data Length is set to 0.This field must be DW aligned, since each coeff icient is 4 bytes in size.Driver will determine the Num of EOB 4x4/8x8 must match the block cbp flags, if not match, hardware cannot hang - add error handling.This field is only valid in AVC, VC1, MPEG2 decoder IT mode. Value: [0,512MB) — Name: (none)

MFD_IT_OBJECT

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

MFD_JPEG_BSD_OBJECT

MFD_JPEG_BSD_OBJECT

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2
Exists If:	//Decoder

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

MFD_JPEG_BSD_OBJECT		
2	31:29	**Reserved** Project: All Format: MBZ
	28:0	**Indirect Data Start Address** Project: All Format: IndirectObjectOffset[28:0] This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This pointer is relative to the BSD Indirect Object Base Address.Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the JPEG bitstream data
3	31:29	**Reserved**
	28:16	**Scan Horizontal Position** Project: All Format: U13 bits in blocks This field indicates the horizontal position (in block units) of the first MCU in the Scan.
	15:13	**Reserved**
	12:0	**Scan Vertical Position** Project: All Format: U13 bits in blocks This field indicates the vertical position (in block units) of the first MCU in the Scan.
4	31	**Reserved** Format: MBZ
	30	**Interleaved** Value / Name / Description: 0 / Non-Interleaved / one component in the Scan 1 / Interleaved / multiple components in the Scan
	29:27	**Scan Components** Bit0: Y Bit1: U Bit2: V For example, if non-interleaved Y, then it will be set to 001b. If interleaved Y, U, and V, it will be set to 111b.
	26	**Reserved** Format: MBZ
	25:0	**MCU Count** Project: All Format: U26 This field indicates the number of MCUs in the Scan.

MFD_JPEG_BSD_OBJECT		
5	31:16	**Reserved** Project: All Format: MBZ
	15:0	**RestartInterval(16 bit)** Project: All Format: U16 Specifies the number of MCU in restart interval. Valid values are 1->0xFFFFValue of 0 implies that all the SCAN have only one ECS.

MFD_MPEG2_BSD_OBJECT

MFD_MPEG2_BSD_OBJECT		
Project:	CHV, BSW	
Source:	VideoCS	
Length Bias:	2	

Different from AVC and VC1, MFD_MPEG2_BSD_OBJECT command is pipelinable. This is for performance purpose as in MPEG2 a slice is defined as a group of MBs of any size that must be within a macroblock row.Slice header parameters are passed in as inline data and the bitstream data for the slice is passed in as indirect data. Of the inline data, slice_horizontal_position and slice_vertical_position determines the location within the destination picture of the first macroblock in the slice. The content in this command is identical to that in the MEDIA_OBJECT command in VLD mode described in the Media Chapter.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFD_MPEG2_BSD_OBJECT Format: OpCode
	26:24	**Media Command Opcode** Default Value: 3h MPEG2_DEC Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h Format: OpCode
	20:16	**SubOpcode B** Default Value: 8h Format: OpCode
	15:12	**Reserved** Project: All Format: MBZ
	11:0	**DWord Length** Default Value: 0003h Excludes DWord (0,1) Project: All Format: =n Total Length - 2

MFD_MPEG2_BSD_OBJECT		
1	31:0	**Indirect BSD Data Length** Project: All Format: U32 It is the length in bytes of the bitstream data for the current slice. It includes the first byte of the first macroblock and the last non-zero byte of the last macroblock in the slice. Specifically, the zero-padding bytes (if present) and the next start-code are excluded. This field is sized to support beyond MPEG-2 MP@HL bitstream (<4K). According to Table 8-6 of ISO/IEC 13818-2, the maximum number of bits per macroblock for 4:2:0 is 4608. So the maximum slice size for 4K x 4K is 4608 * 256 / 8 = 147,456 bytes (0x24000), which requires 18 bits. **Programming Notes** / **Project** As MPEG-2 spec does not post any limitation of the size of zero-padding bytes, it is possible to have a slice data with large length (including zero-padding bytes). As the data beyond 0x10E00 would only be zero bytes for a valid slice data / zero-padding restriction is removed / CHV, BSW
2	31:29	**Reserved** Project: All Format: MBZ
	28:0	**Indirect Data Start Address** Format: IndirectObjectOffset[28:0] This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This pointer is relative to the BSD Indirect Object Base Address.Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the MPEG2 VLD bitstream data This address points to the first byte of the MB layer data, i.e. not including slice header.
3..4	31:0	**Inline Data** All the required Slice Header parameters and error handling settings are captured as MPEG2_BSD_OBJECT Inline Data Descriptor structures. It has a fixed size of 2 DWs. Its definition is described in the next section.

MFD_VC1_BSD_OBJECT

MFD_VC1_BSD_OBJECT

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

The MFD_VC1_BSD_OBJECT command is the only primitive command for the VC1 Decoding Pipeline. The macroblock data portion of the bitstream is loaded as indirect data object.Before issuing a MFD_VC1_BSD_OBJECT command, all VC1 states of the MFD Engine need to be valid. Therefore the commands used to set these states need to have been issued prior to the issue of a MFD_VC1_BSD_OBJECT command.VC1 deblock filter kernel cross the slice boundary if in the last MB row of a slice, so need to know the last MB row of a slice to disable the edge mask. There is why VC1 BSD hardware need to know the end of MB address for the current slice. As such no more phantom slice is needed for VC1, as long as the driver will program both start MB address in the current slice and the start MB address of the next slice. As a result, we can also support multiple picture state commands in between slices.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_MULTI_DW Format: OpCode
	26:24	**Media Command Opcode** Default Value: 2h VC1_DEC Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h Format: OpCode
	20:16	**SubOpcode B** Default Value: 8h Format: OpCode
	15:12	**Reserved** Project: All Format: MBZ
	11:0	**DWord Length** Default Value: 0003h Excludes DWord (0,1) Project: All Format: =n Total Length - 2

MFD_VC1_BSD_OBJECT

1	31:24	**Reserved** Project: All Format: MBZ
	23:0	**Indirect BSD Data Length** Project: All Format: U24 This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. This field must have the same alignment as the Indirect Object Data Start Address.GEN6 Long Format : It is the length in bytes of the bitstream data for the current slice/picture. It includes the first byte of the first macroblock and the last byte of the last macroblock in the slice/picture. Specifically, the zero-padding bytes (if present) and the next start-code are excluded. Hardware ignores the contents after the last non-zero byte (trailing zeros). This field is sized to support VC1 AP@L4 Level bitstream. It includes the byte that contains the First MB Bit OffsetGEN7 Short Format : It is the length in bytes of the bitstream data for the current slice, including Picture/Slice Header + Emulation Prevention Bytes + any filling trailing zeros after the last MB. Hardware ignores the contents after the last non-zero byte. Trailing zero is allowed and handled correctly.
2	31:29	**Reserved**
	28:0	**Indirect Data Start Address** Project: All Format: IndirectObjectOffset[28:0] This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This pointer is relative to the MFD Indirect Object Base Address.Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VC1 bitstream data. Value: [0,512MB) — Name: (none)
3	31:24	**Reserved**
	23:16	**Slice Start Vertical Position** This field specifies the position in y-direction of the first macroblock in the Slice in unit of macroblocks. For SecondField this value is reset to zero as oppoed to the VC1 spec Ref: 9.1.2 Slice Layer.This field is for both Long and Short VC1 Interface Format.
	15:9	**Reserved** Project: All Format: MBZ
	8:0	**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)This field is maintained and provided by the driver for both Long and Short VC1 Interface Format.

MFD_VC1_BSD_OBJECT

<table>
<tr><td>4</td><td>31:16</td><td>First_MB_Byte_Offset of Slice Data or Slice Header
For DXVA2 VC1 Short Format onlyIt gives the byte offset to locate the first MB data in the bitstream for a slice, relative to the Indirect BSD Data Start Address.</td></tr>
<tr><td></td><td>15:5</td><td>Reserved
<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>4</td><td>Emulation Prevention Byte Present
<table><tr><th>Value</th><th>Name</th><th>Description</th><th>Project</th></tr><tr><td>0h</td><td></td><td>H/W needs to perform Emulation Byte Removal</td><td>All</td></tr><tr><td>1h</td><td></td><td>H/W does not need to perform Emulation Byte Removal</td><td>All</td></tr></table></td></tr>
<tr><td></td><td>3</td><td>Reserved
<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>2:0</td><td>FirstMbBitOffset (First Macroblock Bit Offset)
<table><tr><td>Format:</td><td>U3</td></tr></table>This field provides the bit offset of the first macroblock of the Slice in the first byte of the input compressed bitstream.It is used with First_MB_Byte_Offset for non-byte aligned position.</td></tr>
</table>

MFD_VC1_LONG_PIC_STATE

MFD_VC1_LONG_PIC_STATE

Project:	CHV, BSW
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**	
		Project:	All
		Format:	MBZ
	11:0	**DWord Length**	
		Default Value:	0004h Excludes DWord (0,1)
		Project:	All
		Format:	=n Total Length - 2

MFD_VC1_LONG_PIC_STATE

<table>
<tr><td>1</td><td>31:24</td><td>Reserved

Project:	CHV, BSW
Format:	MBZ

</td></tr>
<tr><td></td><td>23:16</td><td>PictureHeightInMBsMinus1 (Picture Height Minus 1 in Macroblocks)

Project:	CHV, BSW
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	Project
[0,254]	Value_0_to_254	a valid range of [0,254] [1, 255] MB	CHV, BSW

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.

</td></tr>
<tr><td></td><td>15:8</td><td>Reserved

Project:	CHV, BSW
Format:	MBZ

</td></tr>
<tr><td></td><td>7:0</td><td>PictureWidthInMBsMinus1 (Picture Width Minus 1 in Macroblocks)

Project:	CHV, BSW
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	Project
[0,254]	Value_0_to_254	[1,255] MB	CHV, BSW

</td></tr>
<tr><td>2</td><td>31:24</td><td>Bitplane Buffer Pitch Minus 1

Project:	CHV, BSW
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]	

</td></tr>
</table>

MFD_VC1_LONG_PIC_STATE

		Programming Notes 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** Project: CHV, BSW Format: MBZ
	15	**DmvSurfaceValid** Project: CHV, BSW 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** Project: CHV, BSW 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** Project: CHV, BSW 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** Project: CHV, BSW 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	Project
0h	Not Present	Sync Marker is not present in the bitstream	CHV, BSW
1h	Maybe present	Sync Marker maybe present in the bitstream	CHV, BSW

MFD_VC1_LONG_PIC_STATE

		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** Project: CHV, BSW 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.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	Project
0XX0b		Chroma Quarter -pel + Luma bicubic. (can only be 1MV)	CHV, BSW
0XX1b		Chroma Half-pel + Luma bicubic. (can be 1MV or 4MV)	CHV, BSW
1XX0b		Chroma Quarter -pel + Luma bilinear. (can only be 1MV)	CHV, BSW
1XX1b		Chroma Half-pel + Luma bilinear	CHV, BSW

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** Project: CHV, BSW 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	Project
0h		Scale down reference picture by factor of 2	CHV, BSW
1h		Scale up reference picture by factor of 2	CHV, BSW

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.

MFD_VC1_LONG_PIC_STATE

	6	**RangeReduction Enable** Project: CHV, BSW 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	Project
0h	Disable	Range reduction is not performed	All
1h	Enable	Range reduction is performed	All

Programming Notes
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	Project
0h	Disable	Disables loop filter	All
1h	Enable	Enables loop filter	All

	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	Project
0h	Disable	to disable overlap smoothing filter	All
1h	Enable	to enable overlap smoothing filter	All

	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**

MFD_VC1_LONG_PIC_STATE

	0	**VC1 Profile** specifies the bitstream profile.This field is used in both VLD and IT modes. Value | Name | Description | Project 0h | Disable | current picture is in Simple or Main Profile (No need to distinguish Simple and Main Profile) | All 1h | Enable | current picture is in Advanced Profile | All **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.
3	31	**Reserved**
	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 | Project 00b | Disable | Progressive Frame Picture | All 01b | Enable | Interlaced Frame Picture | All 10b | | Field Picture with Top Field First | 11b | | Field Picture with Bottom Field First |

<table>
<tr><th colspan="3">MFD_VC1_LONG_PIC_STATE</th></tr>
<tr><td></td><td>23:21</td><td>Reserved<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>20:16</td><td>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.</td></tr>
<tr><td></td><td>15:13</td><td>Reserved<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>12:8</td><td>PQuant (Picture Quantization Value)<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>U5</td></tr></table>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).</td></tr>
<tr><td></td><td>7:0</td><td>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</td></tr>
<tr><td>4</td><td>31:30</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>29:28</td><td>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.</td></tr>
</table>

MFD_VC1_LONG_PIC_STATE

Value	Name	Description	Project
00b		Mixed MV, Q-pel bicubic	All
01b		1-MV, Q-pel bicubic	All
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	Project
0h	Disable	only 1-MV	All
1h	Enable	1, 2, or 4 MVs	All

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	Project
0h		Top (even) field	All
1h		Bottom (odd) field	All

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

MFD_VC1_LONG_PIC_STATE

	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: [0, 15] — Name: (none)
	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.
	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
	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.

ExtendedDMVRange (bits 11:10):

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

ExtendedMVRange (bits 9:8):

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]

MFD_VC1_LONG_PIC_STATE

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

	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	Project
0h		Non-uniform	All
1h		Uniform	All

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.

MFD_VC1_LONG_PIC_STATE

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

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

MFD_VC1_LONG_PIC_STATE

	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. <table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Disable</td><td>Non-Raw Mode</td></tr><tr><td>1h</td><td>Enable</td><td>Raw Mode</td></tr></table>
	23	**Reserved** <table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table>
	22:20	**MvTab (Motion Vector Table)** <table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>U3</td></tr></table>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.

MFD_VC1_LONG_PIC_STATE

	15:14	**Reserved** Project: All Format: MBZ
	13:12	**TransType (Picture-level Transform Type)** Project: All 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.
	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.

MFD_VC1_LONG_PIC_STATE		
	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.
	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.

TransDcTab values:

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

MFD_VC1_SHORT_PIC_STATE

MFD_VC1_SHORT_PIC_STATE		
Project:	CHV, BSW	
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** Project: All Format: MBZ
	11:0	**DWord Length** Default Value: 0003h Excludes DWord (0,1) Project: All Format: =n Total Length - 2
1	31:24	**Reserved** Project: All 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

MFD_VC1_SHORT_PIC_STATE

<table>
<tr><td></td><td></td><td>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.
<table><tr><th>Value</th><th>Name</th><th>Description</th><th>Project</th></tr><tr><td>[0,127]</td><td>Value_0_to_127</td><td>[1, 128] MB</td><td>CHV, BSW</td></tr></table></td></tr>
<tr><td></td><td>15:8</td><td>Reserved
<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td>Picture Width
<table><tr><td>Format:</td><td>U8-1 Picture Width in Macroblocks</td></tr></table>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.
<table><tr><th>Value</th><th>Name</th><th>Description</th><th>Project</th></tr><tr><td>[0,127]</td><td>Value_0_to_127</td><td>[1, 128] MB</td><td>CHV, BSW</td></tr></table></td></tr>
<tr><td>2</td><td>31:24</td><td>Bitplane Buffer Pitch Minus 1
<table><tr><td>Format:</td><td>U7-1 Pitch in Bytes</td></tr></table>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.</td></tr>
<tr><td></td><td>23</td><td>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.</td></tr>
<tr><td></td><td>22:20</td><td>Reserved
<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
</table>

MFD_VC1_SHORT_PIC_STATE

	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 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**
	11	**VC1 Profile** Project: All 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. 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
	10:6	**Reserved** Project: All 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.

MFD_VC1_SHORT_PIC_STATE

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

Project:	All
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

DWord 3

31 **Reserved**

Project:	All
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	Project
0h	Disable	to disable overlap smoothing filter	All
1h	Enable	to enable overlap smoothing filter	All

29 **Range Reduction Scale**

Project:	All
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.

MFD_VC1_SHORT_PIC_STATE

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

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	Project
0h	Disable **[Default]**	Range reduction is not performed	All
1h	Enable	Range reduction is performed	All

27:24 **Reserved**

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	Project
0		progressive only picture	All
1		progressive only picture	All
2		interlace picture (frame-interlace or field-interlace)	
3		illegal	

21 **Reserved**

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

Project:	All
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** <table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>00b</td><td></td><td>implicit quantizer at frame leve</td></tr><tr><td>01b</td><td></td><td>explicit quantizer at frame level, and use PQUANTIZER SE to specify uniform or non-uniform</td></tr><tr><td>10b</td><td></td><td>explicit quantizer, and non-uniform quantizer for all frames</td></tr><tr><td>11b</td><td></td><td>explicit quantizer, and uniform quantizer for all frames</td></tr></table>
	13	**MULTIRES Present Flag (for Simple/Main Profile only)** <table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td></td><td>RESPIC Parameter is present in the picture header</td></tr><tr><td>1h</td><td></td><td>RESPIC Parameter is present in the picture header</td></tr></table>
	12	**SYNCMARKER Present Flag (for Simple/Main Profile only)** <table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td></td><td>Bitstream for Simple and Main Profile has no sync marker</td></tr><tr><td>1</td><td></td><td>Bitstream for Simple and Main Profile may have sync marker(s)</td></tr></table>
	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. <table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td></td><td>Range Reduction Parameter (RANGEREDFRM) is not present in the picture header</td></tr><tr><td>1</td><td></td><td>Range Reduction Parameter (RANGEREDFRM) is present in the picture header.</td></tr></table>
	10:8	**MAXBFRAMES** Number of consecutive B Frames.
	7	**PANSCAN Present Flag** <table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td></td><td>Pan Scan Parameters are not present in the picture header</td></tr><tr><td>1</td><td></td><td>Pan Scan Parameters are present in the picture header</td></tr></table>
	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 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

MFD_VC1_SHORT_PIC_STATE

VLD mode only, not in IT mode.

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

Bit 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	Project
0h		no rounding	All
1h		quarter-pel offsets to half/full pel positions	All

Bit 3

EXTENDED_MV Present Flag

BitFieldDesc

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

Bits 2:1

DQUANT

Project:	All
Access:	None
Format:	U2

Use for Picture Header Parsing of VOPDUANT elements

Value	Name	Description
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	

Bit 0

VSTRANSFORM flag

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

DWord 4, Bits 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** Project: All Format: MBZ Advanced Profile only; RANGE_MAPY_FLAG Range Mapping not supported
	22:20	**Reserved** Project: All Format: MBZ Advanced Profile only; RANGE_MAPY Range Mapping not supported
	19	**Reserved** Project: All Format: MBZ Advanced Profile only; RANGE_MAPUV_FLAG Range Mapping not supported
	18:16	**Reserved** Project: All Format: MBZ Advanced Profile only; RANGE_MAPUV Range Mapping not supported
	15:9	**Reserved** Project: All 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

MFD_VC1_SHORT_PIC_STATE		
		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.
	1	**PSF**
	0	**EXTENDED_DMV Present Flag**

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

Value	Name	Description
0h	**[Default]**	Extended_DMV is not present in the picture header
1h		Extended_DMV is present in the picture header

MFD_VP8_BSD_OBJECT

MFD_VP8_BSD_OBJECT

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

The MFD_VP8_BSD_OBJECT command is the only primitive command for the VP8 Decoding Pipeline. The Partitions of the bitstream is loaded as indirect data object. Before issuing a MFD_VP8_BSD_OBJECT command, all VP8 frame level states of the MFD Engine need to be valid. Therefore the commands used to set these states need to have been issued prior to the issue of a MFD_VP8_BSD_OBJECT command. Context switch interrupt is not supported by this command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFD_VP8_BSD_OBJECT Format: OpCode
	26:24	**Media Command OpCode** Default Value: 4h VP8_DEC Format: OpCode
	23:21	**subOpcodeA** Default Value: 1h Format: OpCode
	20:16	**subOpcodeB** Default Value: 8h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 14h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:21	**Reserved** Format: MBZ
	20:16	**Partition0 CPBAC Entropy Count** Pass the Partition0 CPBAC State to HW. Max value is 24.
	15:8	**Partition0 CPBAC Entropy Range** Pass the Partition0 CPBAC State to HW.

MFD_VP8_BSD_OBJECT		
	7:6	**Reserved** Format: MBZ
	5:4	**Coded Num of Coeff Token Partitions** Num of Partitions = 2^CodedNumCoeffTokenParititons. 0 = 1 Partition only 1 = 2 Partitions 2 = 4 Partitions 3 = 8 Partitions are present in the bitstream.
	3	**Reserved** Format: MBZ
	2:0	**Partition0 FirstMBBitOffset from Frame Header** Allow HW to jump to the location in the bitstream where per MB information starts in the Partition0.
2	31:24	**Partition0 CPBAC Entropy Value** Pass the Partition0 CPBAC State to HW.
	23:0	**Reserved** Format: MBZ
3	31:24	**Reserved** Format: MBZ
	23:0	**Indirect Partition0 Data Length** This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition.
4	31:0	**Indirect Partition0 Data Start Offset** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.
5	31:24	**Reserved** Format: MBZ
	23:0	**Indirect Partition1 Data Length** This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition.
6	31:0	**Indirect Partition1 Data Start Offset** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.
7	31:24	**Reserved**

MFD_VP8_BSD_OBJECT

	23:0	**Indirect Partition2 Data Length** This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition.
8	31:0	**Indirect Partition2 Data Start Offset** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.
9	31:24	**Reserved** Format: MBZ
	23:0	**Indirect Partition3 Data Length** This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition.
10	31:0	**Indirect Partition3 Data Start Offset** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.
11	31:24	**Reserved**
	23:0	**Indirect Partition4 Data Length** This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition.
12	31:0	**Indirect Partition4 Data Start Offset** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.
13	31:24	**Reserved**
	23:0	**Indirect Partition5 Data Length** This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition.

MFD_VP8_BSD_OBJECT		
14	31:0	**Indirect Partition5 Data Start Offset** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.
15	31:24	**Reserved** Format: MBZ
	23:0	**Indirect Partition6 Data Length** This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition.
16	31:0	**Indirect Partition6 Data Start Offset** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.
17	31:24	**Reserved** Format: MBZ
	23:0	**Indirect Partition7 Data Length** This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition.
18	31:0	**Indirect Partition7 Data Start Offset** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.
19	31:24	**Reserved**
	23:0	**Indirect Partition8 Data Length** This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition.
20	31:0	**Indirect Partition8 Data Start Offset** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.

MFD_VP8_BSD_OBJECT

21	31	**Concealment Method** This field specifies the method used for concealment when error is detected. <table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td>Intra 16x16 Prediction</td><td>A copy from the current picture is performed using Intra 16x16 Prediction method.</td></tr><tr><td>1</td><td>Inter P Copy</td><td>A copy from collocated macroblock location is performed from the concealment reference indicated by the ConCeal_Pic_Id field.</td></tr></table>
	30:18	**Reserved** Format: MBZ
	17:16	**Conceal_Pic_Id (Concealment Picture ID)** Exists If: [Concealment Method] == 1 This field identifies the picture in the reference list to be used for concealment. This field is only valid if Concealment Method is Inter P Copy. 00 - Last Decoded Picture 01 - Golden Reference Picture 02 - Alternate Reference Picture 03 - User provided Reference Picture
	15	**Reserved** Format: MBZ
	14	**BSDPrematureComplete Error Handling** It occurs in situation where the decode is completed but there are still data in the bitstream. <table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>Ignore the error and continue (masked the interrupt), assume the hardware automatically perform the error handling</td></tr><tr><td>1</td><td>Set the interrupt to the driver (provide MMIO registers for MB address R/W)??</td></tr></table>
	13	**Reserved** Format: MBZ
	12	**MPR Error (MV out of range) Handling** <table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>Ignore the error and continue (masked the interrupt), assume the hardware automatically perform the error handling</td></tr><tr><td>1</td><td>Set the interrupt to the driver (provide MMIO registers for MB address R/W)??</td></tr></table>
	11	**Reserved**
	10	**Entropy Error Handling** <table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>Ignore the error and continue (masked the interrupt), assume the hardware automatically perform the error handling</td></tr><tr><td>1</td><td>Set the interrupt to the driver (provide MMIO registers for MB address R/W)??</td></tr></table>
	9	**Reserved** Format: MBZ

<table>
<tr><th colspan="3">MFD_VP8_BSD_OBJECT</th></tr>
<tr><td rowspan="2"></td><td>8</td><td>MB Header Error Handling
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0</td><td>Ignore the error and continue (masked the interrupt), assume the hardware automatically perform the error handling</td></tr>
<tr><td>1</td><td>Set the interrupt to the driver (provide MMIO registers for MB address R/W)??</td></tr>
</table></td></tr>
<tr><td>7:0</td><td>Reserved
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
</table>

MFL_AVS_BSD_OBJECT

MFL_AVS_BSD_OBJECT

Project:	CHV, BSW
Source:	PRM
Length Bias:	2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h MFL_AVS_BSD_OBJECT Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 9h MFL_Decoder Format: OpCode
	22:16	**Media Instruction Command** Default Value: 22h MFL_AVS_BSD_OBJECT Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Length -2 Value: 3h — Name: DWORD_COUNT_n **[Default]** — Description: Excludes Dwords 0,1
1	31:24	**Reserved**
	23:0	**Indirect BSD Data Length** It is the length in bytes of the bitstream data for the current slice. It includes the first byte of the first macroblock and the last non-zero byte of the last macroblock in the slice. Specifically, the zero-padding bytes (if present) and the next start-code are excluded.
2	31:29	**Reserved**
	28:0	**Indirect Data Start Address** Format: GraphicsAddress[28:0] Specifies a byte offset to the **BSD Indirect Object Base Address** for the starting address of the slice level bit stream, which begins with the slice Start Code Prefix (00 00 01b). The starting bit of the bit stream is always byte aligned.

MFL_AVS_BSD_OBJECT		
3	31	**Reserved** Format: MBZ
	30:24	**Slice Horizontal Position** This field indicates the horizontal position in macroblock units of the first macroblock in the slice. Value: 0-127 — Name: — Description: Valid range is inclusive.
	23	**Reserved** Format: MBZ
	22:16	**Slice Vertical Position** This field indicates the vertical position in macroblock units of the first macroblock in the slice. Value: 0-127 — Name: — Description: Valid range is inclusive.
	15:8	**Macroblock Count** This field indicates the number of macroblocks in the slice, including skipped macroblocks. MFL ignores this field.
	7:6	**Reserved** Format: MBZ
	5	**Last Pic Slice** This bit is required for error concealment. Value: 0 — Name: — Description: Not the last slice of the picture Value: 1 — Name: — Description: Last slice of picture
	4:3	**Reserved** Format: MBZ
	2:0	**First MB Bit Offset** This field provides the bit offset of the first macroblock in the first byte of the input bit stream.
4	31:29	**Reserved**
	28:24	**Quantizer Scale Code Slice** This field sets the quantizer scale code of the inverse quantizer. It remains in effect until changed by a decoded quantizer scale code in a macroblock. This field is decoded from the slice header by host software.
	23	**Closest Reference Picture** For B pictures, indicates which reference picture is closest in display order to the current picture. This information is used for error concealment when copying the co-located macroblock. Value: 0 — Name: — Description: Backward Reference Frame Value: 1 — Name: — Description: Forward Reference Frame

MFL_AVS_BSD_OBJECT

	22:15	**Reserved** Format: MBZ
	14:8	**Next Slice Vertical Position** This field indicates the vertical position in macroblock units of the first macroblock in the next slice. Value: 0-127 — Name: — Description: The range is inclusive **Programming Notes** 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 the picture (field picture will be in height of field) (since y-direction is zero-based numbering).
	7	**Reserved** Format: MBZ
	6:0	**Next Slice Horizontal Position** This field indicates the horizontal position in macroblock units of the first macroblock in the next slice. Value: 0-127 — Name: — Description: The range is inclusive **Programming Notes** This field is primarily used for error concealment. In the case that current slice is the last slice, this field should set 0.

MFL_AVS_PIC_STATE

MFL_AVS_PIC_STATE

Project:	CHV, BSW
Source:	PRM
Length Bias:	2

This must be the first command to be issued after the surface state, the pipe select and base address commands.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h MFL_AVS_PIC_STATE Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 9h MFL_Decoder Format: OpCode
	22:16	**Media Instruction Command** Default Value: 20h MFL_AVS_PIC_STATE Format: OpCode
	15:12	**Reserved**
	11:0	**DWord Length** Format: =n Length -2 Value | Name | Description Ah | DWORD_COUNT_n **[Default]** | Excludes Dwords 0,1
1	31:26	**Reserved**
	25	**padding_mode** Format: MBZ
	24:23	**chroma_sampling_mode** Programming Notes Must be 1 (for Ycbcr - Digital Sampling of 4:2:0)
	22	**top_field_first** Value | Name 0 | Bottom Field First 1 | Top Field First

MFL_AVS_PIC_STATE

	21	**field_picture** <table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>Frame picture/Progressive</td></tr><tr><td>1</td><td>Field Picture</td></tr></table>
	20	**field_num** <table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>Top Field</td></tr><tr><td>1</td><td>Bottom Field</td></tr></table>
	19	**frame_pred_frame_dct** Format: MBZ
	18:17	**slice_type** Format: MBZ
	16:12	**Reserved** Format: MBZ
	11	**intra_vlc_format** Format: MBZ
	10	**concealment_motion_vectors** Format: MBZ
	9	**FW Reference** Format: MBZ
	8:7	**intra_dc_precision** Format: MBZ
	6	**short_video_header** Format: MBZ
	5	**interlaced** The interlaced flag in the Video Object Layer header. <table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td></td><td>VOPs associated with the VOL are of non-interlaced (or progressive) format</td></tr><tr><td>1</td><td></td><td>VOPs associated with the VOL may contain interlaced video.</td></tr></table> <table><tr><th>Programming Notes</th></tr><tr><td>Must be set to "1" for field pictures or interlaced frame picture for all other standards.</td></tr></table>
	4	**alternate_vertical_scan_flag** Format: MBZ
	3	**quant_type** Format: MBZ

MFL_AVS_PIC_STATE		
	2:0	**Reserved** Format: MBZ
2	31:22	**Reserved** Format: MBZ
	21	**Past Field/Frame** Format: MBZ
	20:16	**Past Frame ID** Format: MBZ
	15:14	**Reserved** Format: MBZ
	13	**Future Field/Frame** Format: MBZ
	12:8	**Future Frame ID** Format: MBZ
	7:6	**Reserved** Format: MBZ
	5	**Current Field/Frame**
	4:0	**Current Frame ID** Format: MBZ
3	31:20	**Reserved** Format: MBZ
	19:16	**Picture Width Mod 16** Specifies the picture width mod 16 in pixels.
	15:7	**Reserved** Format: MBZ
	6:0	**Macroblocks Per Row Minus One** Specifies the number of 16x16 macroblocks in a macroblock row of the video VOP, frame or picture. Value: 0-127 / Name: / Description: The range is inclusive
4	31:20	**Reserved** Format: MBZ
	19:16	**Picture Height Mod 16** Specifies the picture height mod 16 in pixels.
	15:7	**Reserved** Format: MBZ

MFL_AVS_PIC_STATE

DWord	Bit	Description
	6:0	**Macroblock Row Height Minus One** Specifies the number of 16x16 macroblock rows of the VOP, frame or picture. Value: 0-127 — Name: — Description: The range is inclusive
5	31:24	**Reserved** Format: MBZ
	23:16	**Second Forward Reference Picture of Current Picture** Farthest from current picture in display order. Value: 2h — Name: Programming Notes: Please note that this field must be set to "2".
	15:8	**First Forward Reference Picture of Current Picture** Format: Must Be One Closest to current picture in display order.
	7:2	**Reserved** Format: MBZ
	1:0	**Current Picture** Format: MBZ
6	31:24	**Reserved** Format: MBZ
	23:16	**Second Forward Reference Picture of Backward Reference** Farthest from backward reference in display order. Value: 2h — Name: Programming Notes: Please note that this field must be set to "2".
	15:8	**First Forward Reference Picture of Backward Reference** Format: Must Be One Closest to backward reference in display order.
	7:0	**Backward Reference Picture** Format: U8 Programming Notes: Must be 3

MFL_AVS_PIC_STATE			
7	31:24	**picture_type (picture structure for AVS)**	
		Format:	MBZ
	23:16	**top_field**	
	15:8	**first_field_in_display_order**	
	7:0	**picture_reference_flag**	
8	31:16	**Reserved**	
		Format:	MBZ
	15:8	**picture_structure frame buffer ID 0**	
	7:0	**picture_distance frame buffer ID 0**	
9	31:16	**Reserved**	
		Format:	MBZ
	15:8	**picture_structure frame buffer ID 1**	
	7:0	**picture_distance frame buffer ID 1**	
10	31:16	**Reserved**	
		Format:	MBZ
	15:8	**picture_structure frame buffer ID 2**	
	7:0	**picture_distance frame buffer ID 2**	
11	31:16	**Reserved**	
		Format:	MBZ
	15:8	**picture_structure frame buffer ID 3**	
	7:0	**picture_distance frame buffer ID 3**	

MFL_BSP_CFG_STATE

MFL_BSP_CFG_STATE

Project: CHV, BSW
Source: PRM
Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h MFL_BSP_CFG_STATE Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 9h MFL_Decoder Format: OpCode
	22:16	**Media Instruction Command** Default Value: 4h MFL_BSP_CFG_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Length -2 Value: 2h; Name: DWORD_COUNT_n **[Default]**; Description: Excludes Dwords 0,1
1	31:1	**Reserved**
	0	**vSparc Synchronous Reset** Value 0: Normal Operation Value 1: Reset vSparc
2	31	**Reserved**
	30:20	**Reserved**
	19:12	**APB PSel** One hot select for MFL's internal APB Bus PSel line.
	11:0	**APB Write Address**
3	31:0	**APB Write Data**

MFL_DEBLOCKER_CFG_STATE

MFL_DEBLOCKER_CFG_STATE

Project:	CHV, BSW
Source:	PRM
Length Bias:	2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h MFL_DEBLOCKER_CFG_STATE Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 9h MFL_Decoder Format: OpCode
	22:16	**Media Instruction Command** Default Value: 6h MFL_DEBLOCKER_CFG_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Length -2 Value: 20h; Name: DWORD_COUNT_n **[Default]**; Description: Excludes Dwords 0,1
1	31:14	**Reserved** Format: MBZ
	13:8	**Deblock Beta Offset** Programming Notes Must be set to zero.
	7:6	**Reserved** Format: MBZ
	5:0	**Deblock Alpha Offset** Programming Notes Must be set to zero.

MFL_DEBLOCKER_CFG_STATE

<table>
<tr><td>2..33</td><td>31:0</td><td>Deblocker Quantization Map Table
This field represents the quantization parameter table used by the deblocking filter. The table is 6-bits wide and 128 entries deep.
<table>
<tr><th>DWord</th><th>31:24</th><th>23:16</th><th>15:8</th><th>7:0</th></tr>
<tr><td>2</td><td>00b, data[3][5:0]</td><td>00b, data[2][5:0]</td><td>00b, data[1][5:0]</td><td>00b, data[0][5:0]</td></tr>
<tr><td>3</td><td>00b, data[7][5:0]</td><td>00b, data[6][5:0]</td><td>00b, data[5][5:0]</td><td>00b, data[4][5:0]</td></tr>
<tr><td>:</td><td>:</td><td>:</td><td>:</td><td>:</td></tr>
<tr><td>33</td><td>00b, data[127][5:0]</td><td>00b, data[126][5:0]</td><td>00b, data[125][5:0]</td><td>00b, data[124][5:0]</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>MPEG2: Please refer to the Table - MPEG2 Deblocker Quantization Map</td></tr>
<tr><td>MPEG4-2: Please refer to the Table - MPEG4-2 Deblocker Quantization Map</td></tr>
<tr><td>AVS: Deblocker Quantization Map Table ignored for AVS</td></tr>
</table>
</td></tr>
</table>

MFL_DMEM_STATE

MFL_DMEM_STATE

Project:	CHV, BSW
Source:	PRM
Length Bias:	2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h MFL_DMEM_STATE Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 9h MFL_Decoder Format: OpCode
	22:16	**Media Instruction Command** Default Value: 8h MFL_DMEM_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Length -2 Value: 1h; Name: ValidValue_1; Project: CHV, BSW; Exists If: //AVS, //MPEG4-2 **Programming Notes** The programming value of this file excludes Dword 0 and 1
1	31:11	**Reserved** Format: MBZ
	10:0	**DMEM Destination Base address** Format: GraphicsAddress[10:0] This address is the 32-bit aligned location of the first DWord in DMEM. Subsequent Dwords are loaded in sequential order. **Programming Notes** Address bits [1:0] must be set to zero.

MFL_DMEM_STATE		
2 **Project:** CHV, BSW	31:0	**Inline DMEM** Project: CHV, BSW **Programming Notes** There must be one DMEM command issued for every inline DWord.

MFL_FFLS_STATE

MFL_FFLS_STATE

Project: CHV, BSW
Source: PRM
Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h MFL_FFLS_STATE Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 9h MFL_Decoder Format: OpCode
	22:16	**Media Instruction Command** Default Value: 9h MFL_FFLS_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Length -2 Value: 8h — Name: DWORD_COUNT_n **[Default]** — Description: Excludes Dwords 0,1
1	31:1	**Reserved**
	0	**FFLS Instance Select** Value 0 — AC/Intra Prediction Row Buffer FFLS — Parameters in DWord2-DWord25 are applied to the AC/Intra Prediction Row Buffer FFLS. Value 1 — Deblocker Row Buffer FFLS — Parameters in DWord2-DWord25 are applied to the Deblocker Row Buffer FFLS **Programming Notes**: For MPEG2, the AC/Intra Prediction Row Buffer FFLS is not used and thus does not need to be configured.

MFL_FFLS_STATE		
2	31:24	**FFLS packet Length 4** FFLS Instance Select = 0: 00h FFLS Instance Select = 1: 00h
	23:16	**FFLS packet Length 3** FFLS Instance Select = 0: 00h FFLS Instance Select = 1: 00h
	15:8	**FFLS packet Length 2** FFLS Instance Select = 0: 00h FFLS Instance Select = 1: 00h
	7:0	**FFLS packet Length 1** FFLS Instance Select = 0: 08h FFLS Instance Select = 1: 08h
3	31:24	**FFLS Low Watermark 4** FFLS Instance Select = 0: FFh FFLS Instance Select = 1: FFh
	23:16	**FFLS Low Watermark 3** FFLS Instance Select = 0: FFh FFLS Instance Select = 1: FFh
	15:8	**FFLS Low Watermark 2** FFLS Instance Select = 0: FFh FFLS Instance Select = 1: FFh
	7:0	**FFLS Low Watermark 1** FFLS Instance Select = 0: 03h FFLS Instance Select = 1: 03h
4	31:24	**FFLS High Watermark 4** FFLS Instance Select = 0: FFh FFLS Instance Select = 1: FFh
	23:16	**FFLS High Watermark 3** FFLS Instance Select = 0: FFh FFLS Instance Select = 1: FFh
	15:8	**FFLS High Watermark 2** FFLS Instance Select = 0: FFh FFLS Instance Select = 1: FFh
	7:0	**FFLS High Watermark 1** FFLS Instance Select = 0: 04h FFLS Instance Select = 1: 04h

MFL_FFLS_STATE		
5	31:6	**FFLS Fill Base Address for Pair 1** Format: GraphicsAddress[31:6] FFLS Instance Select = 0: **Intra Row Store Scratch Buffer Base Address[31:6]** FFLS Instance Select = 1: **Deblocking Filter Row Store Scratch Buffer Base Address[31:6]**
	5:0	**Reserved**
6	31:16	**FFLS Fill-Spill Buffer Depths for Pair 1** FFLS Instance Select = 0: 0080h FFLS Instance Select = 1: 0100h
	15:0	**FFLS Fill-Spill Buffer Depths for Pair 2** FFLS Instance Select = 0: 0000h FFLS Instance Select = 1: 0000h
7	31:24	**FFLS Fill-Spill Transfer Size for Pair 1** FFLS Instance Select = 0: 01h FFLS Instance Select = 1: 01h
	23:16	**FFLS Fill-Spill Transfer Size for Pair 2** FFLS Instance Select = 0: 00h FFLS Instance Select = 1: 00h
	15:8	**FFLS Fill-Spill Transfer Size for Pair 3** FFLS Instance Select = 0: 00h FFLS Instance Select = 1: 00h
	7:0	**FFLS Fill-Spill Transfer Size for Pair 4** FFLS Instance Select = 0: 00h FFLS Instance Select = 1: 00h
8	31:24	**FFLS Fill-Spill Local Buffer Depth for Pair 1** FFLS Instance Select = 0: 04h FFLS Instance Select = 1: 04h
	23:16	**FFLS Fill-Spill Local Buffer Depth for Pair 2** FFLS Instance Select = 0: 01h FFLS Instance Select = 1: 01h
	15:8	**FFLS Fill-Spill Local Buffer Depth for Pair 3** FFLS Instance Select = 0: 01h FFLS Instance Select = 1: 01h
	7:0	**FFLS Fill-Spill Local Buffer Depth for Pair 4** FFLS Instance Select = 0: 01h FFLS Instance Select = 1: 01h

MFL_FFLS_STATE			
9	31:2	**Reserved**	
		Format:	MBZ
	1	**Fill Spill Enable**	
		FFLS Instance Select = 0: 01h	
		FFLS Instance Select = 1: 01h	
	0	**Single Producer to Multiple Consumer Enable**	
		FFLS Instance Select = 0: 00h	
		FFLS Instance Select = 1: 00h	

MFL_IMEM_STATE

MFL_IMEM_STATE

Project: CHV, BSW

Source: PRM

Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h MFL_IMEM_STATE Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 9h MFL_Decoder Format: OpCode
	22:16	**Media Instruction Command** Default Value: 7h MFL_IMEM_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Length -2 Value: 2h; Name: DWORD_COUNT_n **[Default]**; Description: Excludes Dwords 0,1
1	31:6	**MFL BSP Instruction Code ObjectBase Address** Format: GraphicsAddress[31:6] Specifies the 64 byte aligned memory base address for the BSP vSparc's firmware image. The firmware image is defined as 16K bytes.
	5:0	**Reserved**
2	31:0	**Reserved**
3	31:0	**MFL BSP Instruction Code ObjectBase Memory Address Attribute** **Programming Notes** Memory Attribute changes based on project. Since MFL is currently defeatured, this field will be left as RESERVED for now

MFL_IND_OBJ_BASE_ADDR_STATE

MFL_IND_OBJ_BASE_ADDR_STATE

Project:	CHV, BSW
Source:	PRM
Length Bias:	2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h MFL_IND_OBJ_BASE_ADDR_STATE Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 9h MFL_Decoder Format: OpCode
	22:16	**Media Instruction Command** Default Value: 3h MFL_IND_OBJ_BASE_ADDR_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Length -2 Value: 4h; Name: DWORD_COUNT_n **[Default]**; Description: Excludes Dwords 0,1
1	31:12	**MFL Indirect Bitstream ObjectBase Address** Format: GraphicsAddress[31:12] Specifies the 4K-byte aligned memory base address for the read-only indirect data object pointed in the MFL_XXX_BSD_OBJECT command for fetching (reading) the compressed Slice Data.
	11:0	**Reserved**
2	31:16	**Reserved**
	15:0	**Reserved**

<table>
<tr><th colspan="3">MFL_IND_OBJ_BASE_ADDR_STATE</th></tr>
<tr><td>3

Programming Notes:
Memory Attribute changes based on project. Since MFL is currently defeatured, this field will be left as RESERVED for now</td><td>31:0</td><td>MFL Indirect Bitstream ObjectBase Address Attributes
Project: CHV, BSW</td></tr>
<tr><td rowspan="2">4</td><td>31:12</td><td>MFL Indirect Bitstream ObjectAccess Upper Bound
Format: GraphicsAddress[31:12]
This field specifies the 4K-byte aligned maximum memory address access by the indirect data object in the MFL_XXX_BSD_OBJECT command for the slice bit stream. Indirect data accessed at this address or greater will cause the MFL to stop issuing requests to the GAC and the BSP VLD will then only receive zero's until a slice done is received.
Programming Notes
Setting this field to 0 will cause this range to be ignored by the MFL.</td></tr>
<tr><td>11:0</td><td>Reserved
Format: MBZ</td></tr>
<tr><td rowspan="2">5</td><td>31:16</td><td>Reserved
Format: MBZ</td></tr>
<tr><td>15:0</td><td>Reserved
Project: CHV, BSW
Format: MBZ</td></tr>
</table>

MFL_MPEG2_BSD_OBJECT

MFL_MPEG2_BSD_OBJECT

Project:	CHV, BSW
Source:	PRM
Length Bias:	2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h MFL_MPEG2_BSD_OBJECT Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 9h MFL_Decoder Format: OpCode
	22:16	**Media Instruction Command** Default Value: 32h MFL_MPEG2_BSD_OBJECT Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Length -2 Value: 4h; Name: DWORD_COUNT_n **[Default]**; Description: Excludes Dwords 0,1
1	31:24	**Reserved**
	23:0	**Indirect BSD Data Length** It is the length in bytes of the bitstream data for the current slice. It includes the first byte of the first macroblock and the last non-zero byte of the last macroblock in the slice. Specifically, the zero-padding bytes (if present) and the next start-code are excluded.
2	31:29	**Reserved**
	28:0	**Indirect Data Start Address** Format: GraphicsAddress[28:0] This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This pointer is relative to the **BSD Indirect Object Base Address.** Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the bitstream data This address points to the first byte of the MB layer data, i.e. not including slice header.

MFL_MPEG2_BSD_OBJECT

3	31	**Reserved** Format: MBZ
	30:24	**Slice Horizontal Position** This field indicates the horizontal position in macroblock units of the first macroblock in the slice. Value: 0-127 — Name: — Description: The range is inclusive.
	23	**Reserved** Format: MBZ
	22:16	**Slice Vertical Position** This field indicates the vertical position in macroblock units of the first macroblock in the slice. Value: 0-127 — Name: — Description: The range is inclusive.
	15:8	**Macroblock Count** Format: U8 This field indicates the number of macroblocks in the slice, including skipped macroblocks. Programming Notes: MFL ignores this field.
	7:6	**Reserved** Format: MBZ
	5	**Last Pic Slice** This bit is required for error concealment. Value 0 — Name: — Description: Not the last slice of the picture Value 1 — Name: — Description: Last slice of picture
	4:3	**Reserved** Format: MBZ
	2:0	**First MB Bit Offset** This field provides the bit offset of the first macroblock in the first byte of the input bit stream.
4	31:29	**Reserved** Format: MBZ
	28:24	**Quantizer Scale Code Slice** This field sets the quantizer scale code of the inverse quantizer. It remains in effect until changed by a decoded quantizer scale code in a macroblock. This field is decoded from the slice header by host software.
	23	**Closest Reference Picture** For B pictures, indicates which reference picture is closest in display order to the current picture. This information is used for error concealment when copying the co-located macroblock.

MFL_MPEG2_BSD_OBJECT

		Value / Name / Description: 0 — Backward Reference Frame; 1 — Forward Reference Frame
	22:15	**Reserved** Format: MBZ
	14:8	**Next Slice Vertical Position**
	7	**Reserved** Format: MBZ
	6:0	**Next Slice Horizontal Position**
5	31:2	**Reserved** Format: MBZ
	1:0	**Start**

Value	Name	Description
0		Backward Reference Frame
1		Forward Reference Frame

22:15 Reserved

Format:	MBZ

14:8 Next Slice Vertical Position

This field indicates the vertical position in macroblock units of the first macroblock in the next slice.

Value	Name	Description
0-127		The range is inclusive.

Programming Notes
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 the picture (field picture will be in height of field) (since y-direction is zero-based numbering).

7 Reserved

Format:	MBZ

6:0 Next Slice Horizontal Position

This field indicates the horizontal position in macroblock units of the first macroblock in the next slice.

Value	Name	Description
0-127		The range is inclusive.

Programming Notes
This field is primarily used for error concealment. In the case that current slice is the last slice, this field should set 0.

DWord 5, 31:2 Reserved

Format:	MBZ

1:0 Start

This field tells the BSD to start frame or start a slice within a frame.

Value	Name
11b	Start Frame
10b	Start Slice

MFL_MPEG2_PIC_STATE

MFL_MPEG2_PIC_STATE

Project: CHV, BSW
Source: VideoCS
Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h MFL_MPEG2_PIC_STATE Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 9h MFL_Decoder Format: OpCode
	22:16	**Media Instruction Command** Default Value: 30h MFL_MPEG2_PIC_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Length -2 Value: 4h; Name: DWORD_COUNT_n **[Default]**; Description: Excludes Dwords 0,1
1	31:26	**Reserved** Format: MBZ
	25	**padding_mode** Format: MBZ
	24:23	**chroma_sampling_mode** 00b: Monochrome 01b: 4:2:0 10b-11b: Undefined

MFL_MPEG2_PIC_STATE

	22	**top_field_first** <table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>Bottom Field First</td></tr><tr><td>1</td><td>Top Field First</td></tr></table>
	21	**field_picture** <table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>Frame Picture/Progressive</td></tr><tr><td>1</td><td>Field Picture</td></tr></table>
	20	**field_num** <table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>Top Field</td></tr><tr><td>1</td><td>Bottom Field</td></tr></table>
	19	**frame_pred_frame_dct** Picture coding extension parameter found in the MPEG2 standard.
	18:17	**slice_type** <table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>I Frame</td></tr><tr><td>01b</td><td>P Frame</td></tr><tr><td>10b</td><td>B Frame</td></tr><tr><td>11b</td><td>D Frame</td></tr></table>
	16:12	**Reserved**
	11	**intra_vlc_format** Picture coding extension parameter found in the MPEG2 standard.
	10	**concealment_motion_vector** picture coding extension parameter found in the MPEG2 standard.
	9	**FW Reference** The FW reference (last P/I) <table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>Field Picture</td></tr><tr><td>1</td><td>Frame Picture</td></tr></table>
	8:7	**intra_dc_precision** The intra_dc_precision is a 2-bit value defined in *Table 6-13 of the MPEG2 Standard Specification*.
	6	**short_video_header** <table><tr><td>Format:</td><td>MBZ</td></tr></table>
	5	**interlaced** Set to the progressive_sequence parameter found in the Sequence Extension header of *MPEG2 Specification*.

MFL_MPEG2_PIC_STATE

	4	**alternate_vertical_scan_flag** This is an internal flag in the Video Object Plane header. **Value** | **Name** | **Description** 1 | | when the use of alternate vertical scan for interlaced VOPs is specified.
	3	**quant_type** This is an internal flag in the Video Object Layer header. **Value** | **Name** | **Description** 0 | | When the second inverse method of quantization is specified.
	2:0	**Reserved** Format: MBZ
2	31:16	**Reserved** Exists If: //MPEG1 Format: MBZ
	31:16	**Reserved** Exists If: //MPEG2 Format: MBZ
	15	**full_pel_backward_vector[T=1]** Exists If: //MPEG1
	15:12	**f_code[S=1][T=1]** Exists If: //MPEG2
	14:12	**backward_f_code[T=1]** Exists If: //MPEG1
	11	**full_pel_backward_vector[T=0]** Exists If: //MPEG1
	11:8	**f_code[S=1][T=0]** Exists If: //MPEG2
	10:8	**backward_f_code[T=0]** Exists If: //MPEG1
	7	**full_pel_forward_vector[T=1]** Exists If: //MPEG1
	7:4	**f_code[S=0][T=1]** Exists If: //MPEG2
	6:4	**forward_f_code[T=1]** Exists If: //MPEG1
	3	**full_pel_forward_vector[T=0]** Exists If: //MPEG1

MFL_MPEG2_PIC_STATE

DWord	Bit	Description
	3:0	**f_code[S=0][T=0]** Exists If: //MPEG2
	2:0	**forward_f_code[T=0]** Exists If: //MPEG1
3	31:20	**Reserved** Format: MBZ
	19:16	**Picture Width Mod 16** Specifies the picture width mod 16 in pixels.
	15:7	**Reserved** Format: MBZ
	6:0	**Macroblocks Per Row Minus One** Specifies the number of 16x16 macroblocks in a macroblock row of the video VOP, frame or picture. Value: 0-127 / Name: / Description: Range is inclusive
4	31:20	**Reserved** Format: MBZ
	19:16	**Picture Height Mod 16** Specifies the picture Height mod 16 in pixels.
	15:7	**Reserved** Format: MBZ
	6:0	**Macroblock Row Height Minus One** Specifies the number of 16x16 macroblock rows of the VOP, frame or picture. Value: 0-127 / Name: / Description: Range is inclusive
5	31:2	**Reserved** Format: MBZ
	1	**direct_bw** Exists If: //[Out-oF-Loop Deblocking] Value: 1 / Name: Backward Reference / Description: QP values are read from the backward reference picture Value: 0 / Name: Forward Reference / Description: QP values are read from the forward reference picture
	0	**direct_field** Exists If: //[Out-oF-Loop Deblocking] direct_field selects the field to read QP values from when the current picture is a field and the direct reference is a field.

MFL_MPEG4_BSD_OBJECT

MFL_MPEG4_BSD_OBJECT

Project:	CHV, BSW
Source:	PRM
Length Bias:	2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h MFL_MPEG4_BSD_OBJECT Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 9h MFL_Decoder Format: OpCode
	22:16	**Media Instruction Command** Default Value: 12h MFL_MPEG4_BSD_OBJECT Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Length -2 Value: 3h; Name: DWORD_COUNT_n **[Default]**; Description: Excludes Dwords 0,1
1	31:24	**Reserved** Format: MBZ
	23:0	**Indirect BSD Data Length** It is the length in bytes of the bitstream data for the current VOP. It includes the first byte of the first macroblock and the last non-zero byte of the last macroblock in the VOP. Specifically, the zero-padding bytes (if present) and the next start-code are excluded.
2	31:29	**Reserved** Format: MBZ
	28:0	**Indirect Data Start Address** Format: GraphicsAddress[28:0]

MFL_MPEG4_BSD_OBJECT		
		Specifies a byte offset to the **BSD Indirect Object Base Address** for the starting address of the VOP bit stream. The starting address of the bit stream is byte aligned and represents the first byte of the first macroblock in a VOP. DWord2 of the MFL_DMEM_STATE command, hw_used_bits, is a 3-bit value that specifies the starting bit of the VOP in the first byte of the bit stream buffer. The starting bit is not required to be byte aligned. **Programming Notes** For MPEG4-2 this field should normally be set to zero.
3	31:0	**Reserved** Format: MBZ
4	31:0	**Reserved** Format: MBZ

MFL_MPEG4_PIC_STATE

MFL_MPEG4_PIC_STATE

Project:	CHV, BSW
Source:	PRM
Length Bias:	2

This must be the first command to be issued after the surface state, the pipe select and base address commands.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h MFL_MPEG4_PIC_STATE Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 9h MFL_Decoder Format: OpCode
	22:16	**Media Instruction Command** Default Value: 10h MFL_MPEG4_PIC_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Length -2 Value: 3h; Name: DWORD_COUNT_n **[Default]**; Description: Excludes Dwords 0,1
1	31:26	**Reserved** Format: MBZ
	25	**padding_mode** Value 0: Padding Mode 0 — Reflects the padding required in the MPEG4-2 standard for out of picture motion vectors for right and bottom edges Value 1: Padding Mode 1 — Reflects non-standard padding for out of picture motion vectors for right and bottom edges

MFL_MPEG4_PIC_STATE

	24:23	**chroma_sampling_mode** **Programming Notes** : Must be 1 (for Ycbcr - Digital Sampling of 4:2:0)
	22	**top_field_first** Value | Name 0 | Bottom Field First 1 | Top Field First
	21	**field_picture** Value | Name 0 | Frame Picture/Progressive 1 | Field Picture
	20	**field picture** Format: MBZ
	19	**frame_pred_frame_dct** Format: MBZ
	18:17	**slice_type** Value | Name 00b | I frame 01b | P frame 10b | B frame 11b | Undefined
	16:12	**Reserved** Format: MBZ
	11	**intra_vlc_format** Format: MBZ
	10	**concealment_motion_vectors** Format: MBZ
	9	**FW Reference** Format: MBZ
	8:7	**intra_dc_precision** Format: MBZ
	6	**short_video_header** The short_video_header is an internal flag in the Video Object Layer header which is set to "1" when an abbreviated header format is used for the VOP. The short header format is included to provide forward compatibility with the H.263 standard.

MFL_MPEG4_PIC_STATE

DWord	Bit	Description
	5	**interlaced** The interlaced flag in the Video Object Layer header.

Value	Name	Description
0		VOPs associated with the VOL are of non-interlaced (or progressive) format
1		VOPs associated with the VOL may contain interlaced video.

Programming Notes
Must be set to "1" for field pictures or interlaced frame picture for all other standards.

DWord	Bit	Description
	4	**alternate_vertical_scan_flag** It is an internal flag in the Video Object Plane header.

Value	Name	Description
1		When the use of alternate vertical scan for interlaced VOPs is specified.
0		When the use of alternate vertical scan for interlaced VOPs is NOT specified.

DWord	Bit	Description
	3	**quant_type** It is an internal flag in the Video Object Layer header.

Value	Name	Description
1		When the first inverse quantization method is specified.
0		When the second inverse method of quantization is specified.

DWord	Bit	Description
	2:0	**Reserved** Format: MBZ
2	31:22	**Reserved** Format: MBZ
	21	**Past Field/Frame** Format: MBZ
	20:16	**Past Frame ID** Format: Must Be One
	15:14	**Reserved** Format: MBZ
	13	**Future Field/Frame** Format: MBZ
	12:8	**Future Frame ID**

Value	Name	Description
2		This field must be set to 2

DWord	Bit	Description
	7:6	**Reserved** Format: MBZ
	5	**Current Field/Frame** Format: MBZ

MFL_MPEG4_PIC_STATE

DWord	Bit	Description
	4:0	**Current Frame ID** Format: MBZ
3	31:20	**Reserved** Format: MBZ
	19:16	**Picture Width Mod 16** Specifies the picture width mod 16 in pixels.
	15:7	**Reserved** Format: MBZ
	6:0	**Macroblocks Per Row Minus One** Specifies the number of 16x16 macroblocks in a macroblock row of the video VOP, frame or picture. Value: 0-127 — Name: — Description: The range is inclusive
4	31:20	**Reserved** Format: MBZ
	19:16	**Picture Height Mod 16** Specifies the picture height mod 16 pixels
	15:7	**Reserved** Format: MBZ
	6:0	**Macroblock Row Height** Format: U7-1 Specifies the number of 16x16 macroblock rows of the VOP, frame or picture. Value: 0-127 — Name: — Description: The range is inclusive

MFL_PIPE_BUF_ADDR_STATE

<table>
<tr><th colspan="3">MFL_PIPE_BUF_ADDR_STATE</th></tr>
<tr><td colspan="3">Project: CHV, BSW
Source: PRM
Length Bias: 2</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="6">0</td><td>31:29</td><td>Command Type
Default Value: 3h PARALLEL_VIDEO_PIPE
Format: OpCode</td></tr>
<tr><td>28:27</td><td>Pipeline Type
Default Value: 2h MFL_PIPE_BUF_ADDR_STATE
Format: OpCode</td></tr>
<tr><td>26:23</td><td>Media Instruction Opcode
Default Value: 9h MFL_Decoder
Format: OpCode</td></tr>
<tr><td>22:16</td><td>Media Instruction Command
Default Value: 2h MFL_PIPE_BUF_ADDR_STATE
Format: OpCode</td></tr>
<tr><td>15:12</td><td>Reserved
Format: MBZ</td></tr>
<tr><td>11:0</td><td>DWord Length
Format: =n Length -2
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>20h</td><td>DWORD_COUNT_n [Default]</td><td>Excludes Dwords 0,1</td></tr></table></td></tr>
<tr><td rowspan="2">1</td><td>31:12</td><td>Pre Deblocking Destination Address
Format: GraphicsAddress[31:12]
Specifies the 4K byte aligned frame buffer address for outputting the non-filtered reconstructed YUV picture (i.e. output of final adder in each codec standard, and prior to the deblocking filter unit).
Programming Notes
This field is ignored if PreDeblockOutEnable is set to 0 (disabled).</td></tr>
<tr><td>11:0</td><td>Reserved
Format: MBZ</td></tr>
<tr><td>2</td><td>31:16</td><td>Reserved
Format: MBZ</td></tr>
</table>

MFL_PIPE_BUF_ADDR_STATE		
	15:0	**Reserved** Project: CHV, BSW Format: MBZ
3	31:0	**Pre Deblocking Destination Address Attributes** Project: CHV, BSW **Programming Notes:** Memory Attribute changes based on project. Since MFL is currently defeatured, this field will be left as RESERVED for now
4	31:12	**Post Deblocking Destination Address** Format: GraphicsAddress[31:12] Specifies the 4KB byte aligned frame buffer address for outputting the post-loop filtered reconstructed YUV picture (i.e. output of the deblocking filter unit) **Programming Notes** This field is ignored if **PostDeblockOutEnable** is set to 0 (disabled).
	11:0	**Reserved** Format: MBZ
5	31:16	**Reserved** Format: MBZ
	15:0	**Reserved** Project: CHV, BSW Format: MBZ
6	31:0	**Post Deblocking Destination Address Attributes** Project: CHV, BSW **Programming Notes:** Memory Attribute changes based on project. Since MFL is currently defeatured, this field will be left as RESERVED for now
7	31:6	**Intra Row Store Scratch Buffer Base Address** Format: GraphicsAddress[31:6] This field provides the 64 byte aligned base address of the scratch buffer (read/write) used by the AVC Intra Prediction unit to store MB information of the previous row for processing of each macroblock in the current row. The Intra Row Store buffer must be 64-byte cacheline aligned. Hardware uses the horizontal address of the current macroblock to address the Intra Row Store. This field is ignored in MPEG2 mode.
	5:0	**Reserved** Format: MBZ
8	31:16	**Reserved** Format: MBZ
	15:0	**Reserved** Project: CHV, BSW Format: MBZ

MFL_PIPE_BUF_ADDR_STATE		
9	31:0	**Intra Row Store Scratch Buffer Base Address Attributes** Project: CHV, BSW **Programming Notes:** Memory Attribute changes based on project. Since MFL is currently defeatured, this field will be left as RESERVED for now
10	31:6	**Deblocking Filter Row Store Scratch Buffer Base Address** Format: GraphicsAddress[31:6]
	5:0	**Reserved** Format: MBZ
11	31:16	**Reserved** Format: MBZ
	15:0	**Reserved** Project: CHV, BSW Format: MBZ
12	31:0	**Deblocking Filter Row Store Scratch Buffer Base Address Attributes** Project: CHV, BSW **Programming Notes:** Memory Attribute changes based on project. Since MFL is currently defeatured, this field will be left as RESERVED for now
13	31:6	**Backward Reference Picture Base Address** Format: GraphicsAddress[31:6] Specifies the 4KB byte aligned backward reference picture base address for the motion compensation operation in MPEG4/AVS/MPEG2.
	5:0	**Reserved** Format: MBZ
14	31:16	**Reserved** Format: MBZ
	15:0	**Reserved** Project: CHV, BSW Format: MBZ
15	31:0	**Backward Reference Picture Base Address Attributes** Project: CHV, BSW **Programming Notes:** Memory Attribute changes based on project. Since MFL is currently defeatured, this field will be left as RESERVED for now
16	31:6	**Forward Reference Picture Base Address [0]** Format: GraphicsAddress[31:6] Specifies the 64 byte aligned forward reference picture [0] base address for the motion compensation operation in MPEG4/AVS/MPEG2.

MFL_PIPE_BUF_ADDR_STATE		
	5:0	**Reserved** Format: MBZ
17	31:16	**Reserved** Format: MBZ
	15:0	**Reserved** Project: CHV, BSW Format: MBZ
18	31:0	**Forward Reference Picture Base Address [0] Attributes** Project: CHV, BSW **Programming Notes:** Memory Attribute changes based on project. Since MFL is currently defeatured, this field will be left as RESERVED for now
19	31:6	**Forward Reference Picture Base Address [1]** Format: GraphicsAddress[31:6] Specifies the 64 byte aligned forward reference picture [1] base address for the motion compensation operation in AVS only.
	5:0	**Reserved** Format: MBZ
20	31:16	**Reserved** Format: MBZ
	15:0	**Reserved** Project: CHV, BSW Format: MBZ
21	31:0	**Forward Reference Picture Base Address [1] Attributes** Project: CHV, BSW **Programming Notes:** Memory Attribute changes based on project. Since MFL is currently defeatured, this field will be left as RESERVED for now
22	31:6	**Currently Decoded Picture MBI Base Address** Format: GraphicsAddress[31:6] Specifies the 4KB byte aligned MBI base address for the currently decoded picture.
	5:0	**Reserved** Format: MBZ
23	31:16	**Reserved** Format: MBZ
	15:0	**Reserved** Project: CHV, BSW Format: MBZ

MFL_PIPE_BUF_ADDR_STATE		
24	31:0	**Currently Decoded Picture MBI Base Address Attributes** Project: CHV, BSW **Programming Notes:** Memory Attribute changes based on project. Since MFL is currently defeatured, this field will be left as RESERVED for now
25	31:6	**Backward Reference Picture MBI Base Address** Format: GraphicsAddress[31:6] Specifies the 4KB byte aligned MBI base address for the backward reference picture.
	5:0	**Reserved** Format: MBZ
26	31:16	**Reserved** Format: MBZ
	15:0	**Reserved** Project: CHV, BSW Format: MBZ
27	31:0	**Backward Reference Picture MBI Base Address Attributes** Project: CHV, BSW **Programming Notes:** Memory Attribute changes based on project. Since MFL is currently defeatured, this field will be left as RESERVED for now
28	31:6	**Forward Reference Picture MBI Base Address [0]** Format: GraphicsAddress[31:6] Specifies the 4KB byte aligned MBI base address for the forward reference picture [0].
	5:0	**Reserved** Format: MBZ
29	31:16	**Reserved** Format: MBZ
	15:0	**Reserved** Project: CHV, BSW Format: MBZ
30	31:0	**Forward Reference Picture MBI Base Address [0] Attributes** Project: CHV, BSW **Programming Notes:** Memory Attribute changes based on project. Since MFL is currently defeatured, this field will be left as RESERVED for now
31	31:6	**Forward Reference Picture [1] MBI Base Address** Format: GraphicsAddress[31:6] Specifies the 4KB byte aligned MBI base address for the forward reference picture [1]. Used for AVS only.

MFL_PIPE_BUF_ADDR_STATE		
	5:0	**Reserved** Format: MBZ
32	31:16	**Reserved** Format: MBZ
	15:0	**Reserved** Project: CHV, BSW Format: MBZ
33	31:0	**Forward Reference Picture [1] MBI Base Address Attributes** Project: CHV, BSW **Programming Notes:** Memory Attribute changes based on project. Since MFL is currently defeatured, this field will be left as RESERVED for now

MFL_PIPE_MODE_SELECT

MFL_PIPE_MODE_SELECT

Project:	CHV, BSW
Source:	PRM
Length Bias:	2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h MFL_PIPE_MODE_SELECT Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 9h MFL_Decoder Format: OpCode
	22:16	**Media Instruction Command** Default Value: 0h MFL_PIPE_MODE_SELECT Format: OpCode
	15:12	**Reserved**
	11:0	**DWord Length** Format: =n Length -2 Value | Name | Description 1h | DWORD_COUNT_n **[Default]** | Excludes Dwords 0,1
1	31:10	**Reserved**
	9	**Post Deblocking Output Enable** Format: Enable This field controls the output write for the reconstructed pixels AFTER the deblocking filter. Value | Name 0 | Disable 1 | Enable
	8	**Pre Deblocking Output Enable** Format: Enable This field controls the output write for the reconstructed pixels BEFORE the deblocking filter. Value | Name 0 | Disable 1 | Enable

<table>
<tr><th colspan="3">MFL_PIPE_MODE_SELECT</th></tr>
<tr><td></td><td>7:4</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>3:0</td><td>Standard Select<table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>MPEG2</td></tr><tr><td>1</td><td>MPEG4-2</td></tr><tr><td>2</td><td>AVS</td></tr><tr><td>3</td><td>MPEG1</td></tr><tr><td>4-15</td><td>Reserved</td></tr></table></td></tr>
<tr><td>2</td><td>31:2</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>1</td><td>Workload Event Reset
This field resets to zero the following read only registers accessible over the Message Channel Interface:
MFL Decode Status
MFL Last Position
MFL PMU Status
MFL PMU Luma Cache Miss Counter
MFL PMU Chroma Cache Miss Counter
MFL PMU Frame Decode Active Counter
<table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>Normal</td></tr><tr><td>1</td><td>Reset to Zero</td></tr></table></td></tr>
<tr><td></td><td>0</td><td>Media Cache Invalidate
This field invalidates the contents of the media cache.<table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>Normal</td></tr><tr><td>1</td><td>Invalidate the Media Cache</td></tr></table></td></tr>
</table>

MFL_QM_STATE

MFL_QM_STATE

Project: CHV, BSW
Source: PRM
Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h MFL_QM_STATE Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 9h MFL_Decoder Format: OpCode
	22:16	**Media Instruction Command** Default Value: 5h MFL_QM_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Length -2 Value: 10h; Name: DWORD_COUNT_n **[Default]**; Description: Excludes Dwords 0,1
1	31:2	**Reserved** Format: MBZ
	1:0	**QM Type** Exists If: //MPEG2, MPEG1, MPEG4-2 Format: U2 This field specifies which Quantizer Matrix is loaded. Value 0: MPEG_INTRA_QUANTIZER_MATRIX Value 1: MPEG_NON_INTRA_QUANTIZER_MATRIX Value 2-3: Reserved

MFL_QM_STATE		
2..17	31:0	**Quantizer Matrix** The format of a Quantizer Matrix is an 8x8 matrix in raster order. Each element is an unsigned byte. <table><tr><th>DWord</th><th>31:24</th><th>23:16</th><th>15:8</th><th>7:0</th></tr><tr><td>2</td><td>AC(0,3)</td><td>AC(0,2)</td><td>AC(0,1)</td><td>DC</td></tr><tr><td>3</td><td>AC(0,7)</td><td>AC(0,6)</td><td>AC(0,5)</td><td>AC(0,4)</td></tr><tr><td>4</td><td>AC(1,3)</td><td>AC(1,2)</td><td>AC(1,1)</td><td>AC(1,0)</td></tr><tr><td>5</td><td>AC(1,7)</td><td>AC(1,6)</td><td>AC(1,5)</td><td>AC(1,4)</td></tr><tr><td>:</td><td>:</td><td>:</td><td>:</td><td>:</td></tr><tr><td>17</td><td>AC(7,7)</td><td>AC(7,6)</td><td>AC(7,5)</td><td>AC(7,4)</td></tr></table>

MFL_SURFACE_STATE

MFL_SURFACE_STATE

Project:	CHV, BSW
Source:	PRM
Length Bias:	2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h MFL_SURFACE_STATE Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 9h MFL_Decoder Format: OpCode
	22:16	**Media Instruction Command** Default Value: 1h MFL_SURFACE_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Length -2 Value: 4h; Name: DWORD_COUNT_n **[Default]**; Description: Excludes Dwords 0,1
1	31:0	**Surface Base Address** Format: GraphicsAddress[31:0] This field reserved for 3D surface state compatibility. **This field is ignored by the MFL Decoder hardware.**
2	31:18	**Height Minus 1** This field specifies the height of the Picture in units of pixels. For PLANAR surface formats, this field indicates the height of the Y (luma) plane. **This field is ignored by the MFL Decoder hardware.**

MFL_SURFACE_STATE

	17:4	**Width Minus 1** This field specifies the width of the Picture in units of pixels. For PLANAR surface formats, this field indicates the width of the Y (luma) plane. **This field is ignored by the MFL Decoder hardware.**
	3:2	**Reserved** Format: MBZ
	1:0	**Cr(V)/Cb(U) Pixel Offset V Direction** Specifies the distance to the U/V values with respect to the even numbered Y channels in the V direction. **This field is ignored by the MFL Decoder hardware.**
3	31:28	**Surface Format** Specifies the format of the surface; it is **ignored by the MFL Decoder hardware.**

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

	27	**Interleave Chroma** This field indicates that the chroma fields are interleaved in a single plane rather than stored as two separate planes. This field is only used for PLANAR surface formats. **This field is ignored by the MFL Decoder hardware.**
	26	**Reserved** Format: MBZ
	25:22	**Surface Object Control State** Format: MFL_MEMORY_OBJECT_CONTROL_STATE [CHV, BSW] This 4-bit field is used in various state commands and indirect state objects to define LLC cacheability, graphics data type, and encryption attributes for memory objects.

MFL_SURFACE_STATE

	Bits	Description
	21:20	**Reserved** Format: MBZ
	19:3	**Surface Pitch Minus1** Format: U17-1 This field specifies the surface pitch. **Programming Notes** • For tiled surfaces, the pitch must be a multiple of the tile width. • 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, 262143] -> [128B, 256KB]**
	2	**Half Pitch for Chroma** Format: Enable This field indicates that the chroma plane(s) will use a pitch equal to half the value specified in the Surface Pitch field. This field is only used for PLANAR surface formats. **This field is ignored by the MFL Decoder hardware.**
	1	**Tiled Surface** Format: Boolean This field specifies whether the surface is tiled. **This field is ignored by the MFL Decoder hardware.**
	0	**Tile Walk** Format: 3D_TileWalk This field specifies the type of memory tiling (XMajor or YMajor) employed to tile this surface. See *Memory Interface Functions* for details on memory tiling and restrictions. **This field is ignored by the MFL Decoder hardware.**

Tiled Surface values:

Value	Name	Description
0	FALSE	Linear
1	TRUE	Tiled

Programming Notes (Tiled Surface)

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

MFL_SURFACE_STATE

DWord	Bit	Description
		Value / **Name** / **Description**: 0 = TILEWALK_XMAJOR; 1 = TILEWALK_YMAJOR **Programming Notes** • The corresponding cache(s) must be invalidated before a previously accessed surface is accessed again with an altered state of this bit.
4	31:30	**Reserved**
	29:16	**X Offset for U(Cb) in pixel** Format: U14 Pixel Offset This field specifies the horizontal offset in pixels from the 420 frame buffer address to the start (origin) of the U(Cb) plane or the interleaved UV plane if **Interleave Chroma** is enabled. This field is only used for PLANAR surface formats. **This field is ignored by the MFL Decoder hardware.** **Programming Notes** For PLANAR_420 surface formats, this field must be zero
	15:14	**Reserved**
	13:0	**Y Offset for U(Cb) in pixel** Format: U14 Pixel Row Offset This field specifies the vertical offset in pixels from the **Surface Base Address** to the start (origin) of the U(Cb) plane. This field is only used for PLANAR surface formats with **Interleave Chroma** disabled. **This field is ignored by the MFL Decoder hardware.** **Programming Notes** For PLANAR_420 surface formats, this field must be zero
5	31:30	**Reserved**
	29:16	**X Offset for V(Cr) in pixel** Format: U14 Pixel Offset This field specifies the horizontal offset in pixels from the **Surface Base Address** to the start (origin) of the V(Cr) plane. This field is only used for PLANAR surface formats with **Interleave Chroma** disabled. **This field is ignored by the MFL Decoder hardware.** **Programming Notes** For PLANAR_420 surface formats, this field must be zero

MFL_SURFACE_STATE		
	15	**Reserved** Format: MBZ
	14:0	**Y Offset for V(Cr) in pixel** Format: U15 Pixel Row Offset This field specifies the vertical offset in pixels from the **Surface Base Address** to the start (origin) of the V(Cr) plane. This field is only used for PLANAR surface formats with **Interleave Chroma** disabled. **This field is ignored by the MFL Decoder hardware.** **Programming Notes** For PLANAR_420 surface formats, this field must be zero

MFX_AVC_DIRECTMODE_STATE

MFX_AVC_DIRECTMODE_STATE

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

This is a picture level command and is issued once per picture. All DMV buffers are treated as standard media surfaces, in which the lower 6 bits are used for conveying surface states.Current Pic POC number is assumed to be available in POCList[32 and 33] of the MFX_AVC_DIRECTMODE_STATE Command.This command is only valid in the AVC decoding in VLD and IT modes, and AVC encoder mode. The same command supports both Long and Short DXVA2 AVC Interface. The DMV buffers are not required to be programmed for encoder mode.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_SINGLE_DW Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h AVC_COMMON Format: OpCode
	23:21	**SubOpcodeA** Default Value: 0h Format: OpCode
	20:16	**SubOpcodeB** Default Value: 2h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 0045h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:6	**Direct MV Buffer Base Address for Picture 0 (In Frame)** Format: GraphicsAddress[31:6] **Note:**This filed is changed to one per frame (both top and bottom field share the same Direct MV Buffer Base Address). This field provides the base address of the DMV write buffer to store motion vectors

MFX_AVC_DIRECTMODE_STATE

		decoded in the current picture (top field), which may be used later as a collocated motion information read buffer of the associated reference picture in decoding subsequent B-pictures that have MB coded in direct mode. It is a private buffer used by the MPR hardware only. Itscontent is not accessed by software. This buffer must be 64-byte cacheline aligned. The write buffer size is 557,056 bytes for 1 frame. Scalable with frame height, but do not scale with frame width as the hardware assumes frame width (in MBs) fixed at 128 (smallest power of 2 value larger than 120 - 1920x1088 screen resolution) It is only valid if the current picture is a progressive frame, MbAff frame, or a top field. There are a total of 32 reference picture (previously decoded) Direct MV Buffers (0 to 31, not including the DMV write buffer 32 and 33 of the current picture) to read in the corresponding collocated DMV and motion information. For reference picture, these 32 DMV read Buffers can be indexed by the frame_store_ID[4:0], which is obtained from RefPicList L0/L1[RefPicIdx]. frame_Store_IDbit[0] (indicator for Top/Bottiom Field). For writing out motion information during the decoding of the current picture, all 34 DMV buffers can be addressed by [img_dec_fs_idc[4:0]«1 + img_structure[1]].
	5:0	**Reserved** Project: CHV, BSW Format: MBZ
2	31:16	**Reserved** Project: CHV, BSW Format: MBZ Reserved for 64-bit address extension.
	15:0	**Direct MV Buffer Base Address for Picture 0 - Read/Write [47:32]** Project: CHV, BSW Description / Project: This field is for the upper range of AACS Bit Vector Surface Starting Byte Address. / This field is used for 48-bit addressing. / CHV, BSW
3..32	63:48	**Reserved** Format: MBZ
	47:32	**Direct MV Buffer Base Address for Reference Frame 1 to 15 (In Frame) High** Project: CHV, BSW Format: GraphicsAddress[47:32] This field is for the upper range of Pre-Deblocking Destination Address. This field is ignored if PreDeblockOutEnable is set to 0 (disable). This field is used for 48-bit addressing.
	31:6	**Direct MV Buffer Base Address for Reference Frame 1 to 15 (In Frame)** Format: GraphicsAddress[31:6]

MFX_AVC_DIRECTMODE_STATE

		Note: This field is changed to one per frame (both top and bottom field shared the same Direct MV Buffer Base Address) This field provides the base address of the DMV buffer for reference frame 2 to 31. They are needed if the current B-Picture has MBs coded in direct mode. It is a private buffer used by the MPR hardware only. Its content is not accessed by software. All these buffers must be 64-byte cacheline aligned. There are a total of 32 possible Direct MV Read Buffers (not including the current write buffer of the current picture) to read in the corresponding DMV. Each read buffer size is 557,056 bytes for 1 frame (the selected colPic). Scalable with frame height, but do not scale with frame width as the hardware assumes frame width (in MBs) fixed at 128 (smallest power of 2 value larger than 120 - 1920x1088 screen resolution). The adjacent DMV buffers are paired ([2 and 3], [4 and 5], [N and N+1], ..[30 and 31]).
	5:0	**Reserved** Format: MBZ Reserved for 64-bit address extension.
33	31:15	**Reserved** Project: CHV, BSW Format: MBZ
	14:13	**Reserved** Project: CHV, BSW Format: MBZ
	12:11	**Reserved** Project: CHV, BSW Format: MBZ
	10:9	**Reserved** Project: CHV, BSW Format: MBZ
	8:7	**Direct MV Buffer Base Address for Reference Frame - Arbitration Priority Control** Project: CHV, BSW 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
	6:5	**Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for Direct MV Buffer for Reference Picture 0 to 15**

MFX_AVC_DIRECTMODE_STATE

DWord	Bit	Description
		Project: CHV, BSW This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream. **Value** / **Name** 00b: Use Cacheability Controls from page table 01b: Uncacheable (UC) - non-cacheable 10b: Writethrough (WT) 11b: Writeback (WB)
	4:3	**Target Cache (TC) for Direct MV Buffer for Reference Picture 0 to 15** Project: CHV, BSW This field allows the choice of LLC vs eLLC for caching **Value** / **Name** 00b: eLLC Only - not snooped in GT 01b: LLC Only 10b: LLC/eLLC Allowed 11b: L3, LLC, eLLC Allowed
	2	**Encrypted Data for Direct MV Buffer for Reference Picture 0 to 15** Project: CHV, BSW Format: Enable This field controls whether data is decrypted while being read. This field is ignored for writes.
	1:0	**Age for QUADLRU (AGE) for Direct MV Buffer for Reference Picture 0 to 15** Project: CHV, BSW This field allows the selection of AGE parameter for a given surface in LLC or eLLC. **Value** / **Name** 11b: Good chance of generating hits. 10b: Next good chance of generating hits 01b: Decent chance of generating hits 00b: Poor chance of generating hits
34	31:6	**Direct MV Buffer Base Address for Write (Write-Only Buffer)(in frame)** Format: GraphicsAddress[31:6] This field provides the base address of the DMV write-only buffer for the current decoding frame/field. It is a private buffer used by the MPR hardware only. Its content is not accessed by software. All these buffers must be 64-byte cacheline aligned, i.e. the same as the above DMV read/write buffers. These 2 buffers can only be addressed by [img_dec_fs_idc[4:0]«1 + img_structure[1]] for the current picture being decoded.

MFX_AVC_DIRECTMODE_STATE

DWord	Bit	Description
		Each write buffer size is 557,056 bytes for 1 frame (the selected colPic). Scalable with frame height, but do not scale with frame width as the hardware assumes frame width (in MBs) fixed at 128 (smallest power of 2 value larger than 120 - 1920x1088 screen resolution). DMV write buffer 32 is valid only if the current picture is a progressive frame, MbAff frame, or a top field. DMV write buffer 33 is valid only if the current picture is a bottom field.
	5:0	**Reserved** Project: CHV, BSW Format: MBZ
35	31:16	**Reserved** Project: CHV, BSW Format: MBZ Reserved for 64-bit address extension.
	15:0	**Direct MV Buffer Base Address for Write (Write-Only Buffer)(in frame) High** Project: CHV, BSW Format: GraphicsAddress[47:32] This field is for the upper range of Direct MV Buffer Base Address. This field is ignored if PreDeblockOutEnable is set to 0 (disable). This field is used for 48-bit addressing.
36	31:15	**Reserved** Project: CHV, BSW Format: MBZ
	14:13	**Reserved** Project: CHV, BSW Format: MBZ
	12:11	**Reserved**
	10:9	**Reserved** Project: CHV, BSW Format: MBZ
	8:7	**Direct MV Buffer Base Address for Write - Arbitration Priority Control** Project: CHV, BSW 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

MFX_AVC_DIRECTMODE_STATE

	6:5	**Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for Direct MV Buffer for Write** Project: CHV, BSW This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream. **Value** / **Name** 00b: Use Cacheability Controls from page table 01b: Uncacheable (UC) - non-cacheable 10b: Writethrough (WT) 11b: Writeback (WB)
	4:3	**Target Cache (TC) for Direct MV Buffer for Write** Project: CHV, BSW This field controls the L3$, LLC and eLLC (eDRAM) cacheability for a given surface. Setting of "00" points to PTE settings which defaults to eDRAM (when present). If no eDRAM, the access will be allocated to LLC. Setting of "01", allocates into LLC and victimizes the line to eDRAM. Setting of "10" allows the line to be allocated in either LLC or eDRAM. Setting of "11" is the only option for a memory access to be allocated in L3$ as well as LLC/eLLC 00b: eLLC Only *("00" setting points TC selection to PTE which defaults to eLLC)* 01b: LLC Only (*Works at the allocation time, later victimization from LLC downgrades the line to eLLC if present*). 10b: LLC/eLLC Allowed. 11b: L3, LLC, eLLC Allowed. **Errata CHV:A-E (FIXED BY:G0 Stepping):** For all system that does NOT use SVM (i.e. coherent L3$ surfaces), back snoops from LLC has to be disabled (**Dis_GtCvUpdtOnRd = "1"**). Than target Cache settings can be programmed as POR requirements of L3/LLC/eDRAM caching. For all systems that does use SVM (i.e. coherent L3$ surfaces), the recomended setting would be "00" in target cache settings. In case of L3 surfaces, the performance has to be tuned between "00" and "11" setting based on the benefits of L3 caching outweighting the degradation of backsnoops. Post G0-stepping, the above w/a for coherent L3$ surfaces is not needed. **Value** / **Name** 00b: eLLC Only - not snooped in GT 01b: LLC Only 10b: LLC/eLLC Allowed 11b: L3, LLC, eLLC Allowed
	2	**Encrypted Data for Direct MV Buffer for Write** Project: CHV, BSW Format: Enable This field controls whether data is decrypted while being read. This field is ignored for writes.

MFX_AVC_DIRECTMODE_STATE		
	1:0	**Age for QUADLRU (AGE) for Direct MV Buffer for Write** Project: CHV, BSW This field allows the selection of AGE parameter for a given surface in LLC or eLLC. If a particular allocation is done at youngest age ("0,1,2") it tends to stay longer in the cache. This option is given to GFX software to be able to decide which surfaces are more likely to generate HITs, hence need to be replaced least often in caches. **Value** / **Name** 11b: Good chance of generating hits. 10b: Next good chance of generating hits 01b: Decent chance of generating hits 00b: Poor chance of generating hits
37..70	31:0	**POC List, POCList[34][31:0]** Each POC value is a signed 32-bit number. One-to-one correspondence with the 34 Direct MV Buffer Address for Reference and Currrent Frames/Fields There are 34 POC entries in the list. For reference picture, only the lower 32 POC [0-31] entries can be used, and POCList[ ] is indexed by the frame_store_ID[4:0], which is obtained from RefPicList L0/L1[RefPicIdx]. frame_Store_IDbit[0] (indicator for Top/Bottiom Field). For current picture, all 34 POC entries [0-33] can be addressed by POCList[img_dec_fs_idc[4:0]«1 + img_structure[1]]. For frame-only mode, every other entry is skipped. For MBAFF and field-only picture, each entry is a field POC, and every two entries are paired.

MFX_AVC_IMG_STATE

MFX_AVC_IMG_STATE

Project:	CHV, BSW
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
	15:0	**Frame Size** Format: U16-1 in MB unit

MFX_AVC_IMG_STATE		
		The value for FrameSizeInMBs must match the product of FrameWidthInMBs and FrameHeightInMBs.Max. Screen resolution is therefore limited to 256 x 256 in MB unit. It is enough to cover all the Profile-Level specified in the current HD-DVD specification. E.g., for 1920x1080, FrameSizeInMBs[15:0] = 8160 (1920/16 * 1088/16; rounded up 1080). This parameter is specified for Intel interface only, not present in the DXVA. **Value** | **Name** | **Description** [0,16383] | | representing Number of MBs [1,16384]
2	31:24	**Reserved** 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** | **Name** | **Description** [0,255] | | 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 PicWidthInMbs = FrameWidthInMbs / (1 + field_pic_flag) internally done. For MBAFF, PicWidthInMapUnits is in MB pair unit, so the bitstream sends only half frame width.

MFX_AVC_IMG_STATE

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

DWord 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 Reserved

Project:	CHV, BSW
Format:	MBZ

12 Weighted_Pred_Flag

Format:	Enable

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

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

Value	Name	Description

MFX_AVC_IMG_STATE

DWord	Bit	Description
		<table><tr><td>0</td><td>DEFAULT **[Default]**</td><td>Specifies that the default weighted prediction is used for B slices</td></tr><tr><td>1</td><td>EXPLICIT</td><td>Specifies that explicit weighted prediction is used for B slices</td></tr><tr><td>2</td><td>IMPLICIT</td><td>Specifies that implicit weighted prediction is used for B slices.</td></tr><tr><td>3</td><td>Reserved</td><td>Illegal value</td></tr></table> <table><tr><th>Programming Notes</th></tr><tr><td>This field must set to 0 for P and I pictures.</td></tr></table>
	9:8	**ImgStruct - Image Structure, img_structure[1:0]** The current encoding picture structure can only takes on 3 possible values <table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>Frame Picture</td></tr><tr><td>01b</td><td>Top Field Picture</td></tr><tr><td>11b</td><td>Bottom Field Picture</td></tr><tr><td>10b</td><td>Invalid, not allowed.</td></tr></table> <table><tr><th>Programming Notes</th></tr><tr><td>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).</td></tr></table>
	7:0	**Reserved** <table><tr><td>Format:</td><td>MBZ</td></tr></table>
4	31:16	**MinFrameWSize** <table><tr><td>Default Value:</td><td>0h</td></tr><tr><td>Format:</td><td>U16</td></tr></table> **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.

MFX_AVC_IMG_STATE

	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. In VDenc mode this must be set to zero as no MB level rate control is used.
	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.
	13	**Reserved**
	12	**MvUnpackedFlag** MVUnPackedEnable (Encoder Only)This field is reserved in Decode mode.
	11:10	**ChromaFormatIdc** Chroma Format IDC, ChromaFormatIdc[1:0]It specifies the sampling of chroma component (Cb, Cr) in the current picture as follows :

MbStatEnabled values:

Value	Name	Description
0	Disable	Disable Reading of Macroblock Status Buffer
1	Enable	Enable Reading of Macroblock Status Buffer

LoadSlicePointerFlag values:

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

MvUnpackedFlag values:

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

ChromaFormatIdc values:

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

MFX_AVC_IMG_STATE

corresponding Monochrome Flag (monochrome_flag) can be derived from this field.

9 **Reserved**

8 **MbMvFormatFlag**

Use MB level MvFormat flag (Encoder Only)

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.

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.

MFX_AVC_IMG_STATE

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

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.

MFX_AVC_IMG_STATE

		<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td>FALSE</td><td>not in MBAFF mode</td></tr><tr><td>1</td><td>TRUE</td><td>in MBAFF mode</td></tr></table>
	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.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>FRAME</td><td>a slice of a coded frame</td></tr><tr><td>1h</td><td>FIELD</td><td>a slice of a coded field</td></tr></table>
5 [ExistsIf]Encode Only	31	**Trellis Quantization Enabled (TQEnb)**<table><tr><td>Format:</td><td>Enable</td></tr></table>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.<table><tr><th>Value</th><th>Name</th><th>Description</th><th>Project</th></tr><tr><td>0h</td><td>Disable</td><td>Use Normal</td><td></td></tr><tr><td>1h</td><td>Enable</td><td>Use Trellis quantization</td><td>CHV, BSW</td></tr></table>
	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.<table><tr><th>Value</th><th>Name</th><th>Description</th><th>Project</th></tr><tr><td>000b</td><td></td><td>Add 1/8</td><td>CHV, BSW</td></tr><tr><td>001b</td><td></td><td>Add 2/8</td><td>CHV, BSW</td></tr><tr><td>010b</td><td></td><td>Add 3/8</td><td>CHV, BSW</td></tr><tr><td>011b</td><td></td><td>Add 4/8 (rounding 0.5)</td><td>CHV, BSW</td></tr><tr><td>100b</td><td></td><td>Add 5/8</td><td>CHV, BSW</td></tr><tr><td>101b</td><td></td><td>Add 6/8</td><td>CHV, BSW</td></tr><tr><td>110b</td><td>Default</td><td>Add 7/8 (Default rounding 0.875)</td><td>CHV, BSW</td></tr></table>
	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.<table><tr><th>Value</th><th>Name</th><th>Description</th><th>Project</th></tr><tr><td>0h</td><td></td><td>Enable Trellis Quantization chroma</td><td>CHV, BSW</td></tr><tr><td>1h</td><td>Default</td><td>Disable Trellis Quantization chroma</td><td>CHV, BSW</td></tr></table>

MFX_AVC_IMG_STATE

26:21 Reserved

Project:	CHV, BSW
Format:	MBZ

20:17 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

Project:	CHV, BSW
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

In VDenc mode, this field must be zero as frame level rate control is used.

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

MFX_AVC_IMG_STATE

	7	**Intra/InterMbIpcmFlag - ForceIPCMControlMask** This field is to Force **IPCM** for Intra or Inter Macroblock size conformance mask.

Value	Name	Description	Project
0h	Disable	Do not change intra or Inter macroblocks even	CHV, BSW
1h	Enable	Change intra or Inter macroblocks MB_type to IPCM	CHV, BSW

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.

	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.

MFX_AVC_IMG_STATE

		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	31:1	**Reserved**
	0	**VSL Top MB Trans8x8flag** Project: CHV, BSW **Value** | **Name** | **Description** 0 | Disable **[Default]** | VSL will only fetch the current MB data. 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

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

MFX_AVC_IMG_STATE

		and above 1/ 4This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between one-eighth and quarter the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of [(FrameBitRateMin- FrameBitRateMinDelta»2), (FrameBitRateMin- FrameBitRateMinDelta»3)).
	15:8	**SliceDeltaQpMin[1]** Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP for bit-count below FrameBitRateMin- below 1/4 and above 1/ 2This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between quarter and half the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of [(FrameBitRateMin- FrameBitRateMinDelta»1), (FrameBitRateMin- FrameBitRateMinDelta»2)).
	7:0	**SliceDeltaQpMin[0]** Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice Level Delta QP for bit-count below FrameBitRateMin - below 1/ 2This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is below FrameBitRateMin by more than half the distance of FrameBitRateMinDelta , i.e., in the range of [0, (FrameBitRateMin- FrameBitRateMinDelta»1).
10 [ExistsIf]Encode Only	31	**FrameBitrateMaxUnit** This field is the Frame Bitrate Maximum Limit Units. Value 0 – Byte – FrameBitRateMax is in units of 32 Bytes when FrameBitrateMaxUnitMode is 1 and in units of 128 Bytes if FrameBitrateMaxUnitMode is 0 Value 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 0h – compatibility mode – FrameBitRateMaxUnit is in old mode (128b/16Kb) Value 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

MFX_AVC_IMG_STATE

DWord	Bit	Description
		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.

DWord	Bit	Description
	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

DWord	Bit	Description
	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)

DWord	Bit	Description
	13:0	**FrameBitRateMin** RangeThe programmable range 0-512KB When FrameBitrateMinUnit is in 0.Programmable range is 0-8190 KB when FrameBitrateMinUnit is in 1.This field is the Frame Bitrate Minimum Limit ()This field along with FrameBitrateMinUnit determines minimum allowed bits in a Frame before Multi-Pass gets triggered (when enabled). In other words, multi-pass is triggered when the actual frame byte count is less than this value. When FrameBitrateMinUnitMode is 0 (compatibility mode) bits 0:11 should be used, bits 12 and 13 should be 0.
11 [ExistsIf]Encode Only	31	**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]**	

<table>
<tr><th colspan="3">MFX_AVC_IMG_STATE</th></tr>
<tr><td></td><td>15</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>14:0</td><td>FrameBitRateMinDelta<table><tr><td>Range: The programmable range 0-1024KB When FrameBitrateMinUnit is in 32Bytes.Programmable range is 0-16380KB when FrameBitrateMinUnit is in 4Kbytes.</td></tr><tr><td>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.</td></tr></table></td></tr>
<tr><td>12</td><td>31:21</td><td>Reserved</td></tr>
<tr><td></td><td>20</td><td>VMD Error Logic<table><tr><td>Project:</td><td>CHV, BSW</td></tr></table><table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td>Disable [Default]</td><td></td></tr><tr><td>1</td><td>Enable</td><td>Error Handling</td></tr></table></td></tr>
<tr><td></td><td>19</td><td>Reserved</td></tr>
<tr><td></td><td>18</td><td>VAD Error Logic<table><tr><td>Project:</td><td>CHV, BSW</td></tr></table><table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td>Enable [Default]</td><td>Error reporting ON in case of premature Slice done</td></tr><tr><td>1</td><td>Disable</td><td>CABAC Engine will auto decode the bitstream in case of premature slice done.</td></tr></table></td></tr>
<tr><td></td><td>17</td><td>Reserved</td></tr>
<tr><td></td><td>16</td><td>Reserved</td></tr>
<tr><td></td><td>15:0</td><td>Reserved</td></tr>
<tr><td>13</td><td>31:30</td><td>Reserved</td></tr>
<tr><td></td><td>29</td><td>Current Picture Has Performed MMCO5
Set to 1 if the current Pic has performed the memory_management_control_operation = = 5.</td></tr>
<tr><td></td><td>28:24</td><td>Number of Reference Frames<table><tr><td>Format:</td><td>U5</td></tr></table><table><tr><td>Range: Range 0 to MaxDpbSize (=16 for Level 4.1)</td></tr><tr><td>Specifies the maximum number of reference frames (frames, field pairs, unpaired field) existed in the current DBP for decoding the current picture.</td></tr></table></td></tr>
</table>

MFX_AVC_IMG_STATE

	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** | **Name** [0,31] |
	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** | **Name** [0,31] |
	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.

MFX_AVC_IMG_STATE

	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**
	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.

<table>
<tr><th colspan="3">MFX_AVC_IMG_STATE</th></tr>
<tr><td rowspan="8">16

[ExistsIf]: Short Format only</td><td>31</td><td>Inter View Order Disable
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Exists If:</td><td>//Short Format Only</td></tr></table>
It indicates how to append inter-view picture into initial sorted reference list. (due to ambiguity in the MVC Spec)
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Default [Default]</td><td>View Order Ascending</td></tr><tr><td>1h</td><td>Disable</td><td>View ID Ascending</td></tr></table></td></tr>
<tr><td>30:22</td><td>Reserved
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>21:18</td><td>Max View IDXL1
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Exists If:</td><td>//Short Format Only</td></tr></table>
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</td></tr>
<tr><td>17:16</td><td>Reserved
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>15:12</td><td>Max View IDXL0
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Exists If:</td><td>//Short Format Only</td></tr></table>
Reference ListL0 It indicates the maximum number of inter-view picture for Reference List L0</td></tr>
<tr><td>11:10</td><td>Reserved
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>9:0</td><td>Current Frame View ID
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Exists If:</td><td>//Short Format Only</td></tr></table>
It indicates the View ID of the current decoding frame</td></tr>
</table>

MFX_AVC_REF_IDX_STATE

MFX_AVC_REF_IDX_STATE

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

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

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

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

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_AVC_REF_IDX_STATE Format: OpCode
	26:24	**Command Opcode** Default Value: 1h AVC Format: OpCode
	23:21	**SubOpcodeA** Default Value: 0h MFX_AVC_REF_IDX_STATE Format: OpCode

<table>
<tr><th colspan="3">MFX_AVC_REF_IDX_STATE</th></tr>
<tr><td></td><td>20:16</td><td>SubOpcodeB
<table><tr><td>Default Value:</td><td>4h MFX_AVC_REF_IDX_STATE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:12</td><td>Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11:0</td><td>DWord Length
<table><tr><td>Default Value:</td><td>0008h</td></tr><tr><td>Format:</td><td>=n</td></tr></table>
Excludes DWords 0,1</td></tr>
<tr><td>1</td><td>31:1</td><td>Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>0</td><td>RefPicList Select
Num_ref_idx_l1_active is resulted from the specifications in both PPS and Slice Header for the current slice. However, since the full reference list L0 and/or L1 are always sent, only present flags are specified instead.
This parameter is specified for Intel interface only, not present in the DXVA.
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>RefPicList 0</td><td>The list that followed represents RefList L0 (Decoder VLD mode) or Ref Idx Mapping Table L0 (Encoder PAK mode)</td></tr>
<tr><td>1</td><td>RefPicList1</td><td>The list that followed represents RefList L1 (Decoder VLD mode) or Ref Idx Mapping Table L1 (Encoder PAK mode)</td></tr></table></td></tr>
<tr><td>2..9</td><td>31:0</td><td>Reference List Entry
This set of fields is always present whenever this command is issued.
It always specifies the full 32 reference pictures in the selected list, regardless they are "existing picture" or not. If a picture is non-existing, the corresponding entry should be set to all ones.
Each list entry is 1 byte. A 32-bit DW can hold 4 list entries in the following format
31:24 entry X+3 (e.g. listY_3)
23:16 entry X+2 (e.g. listY_2)
15:8 entry X+1 (e.g. listY_1)
7:0 entry X (e.g. listY_0)
X is replaced by the paddr[2:0] * 4 ; paddr[5:0] with 0x20 and 0x27, and Y is replaced by 0 or 1.
The byte definition for a reference picture :
Bit 7 : Non-Existing - indicates that frame store index that should have been at this entry did not exist and was replaced by an index 0 (a valid entry) for error concealment
Bit 6 : Long term bit - set this reference picture to be used as long term reference
Bit 5 : Field picture flag - indicates frame/field
Bit 4:0 : Frame store index or Frame Store ID (Bit 4:1 is used to form the binding table index in intel implementation)</td></tr>
</table>

MFX_AVC_REF_IDX_STATE		
		This is the final Reference List L0 or L1 after any reordering specified in the Slice Header as well as modified by the driver, and its indices values are all translated to the intel specification. If the reference picture is a frame (Bit5 = 1), frame store ID is always an even number. This list is used in outputting MV information by the BSD unit in VLD mode. DMV access also reads and writes Mvlist0 using this frame store ID. If this set of fields is interpreted as Reference Index Mapping Table L0/L1, the same field alignment is followed, i.e. 4 mapping entries per DW. Each mapping entry is one byte in size, but only the least significant 5 bits [4:0] is relevant. Driver should zero all the upper bits [7:5] for each entry.

MFX_AVC_SLICE_STATE

MFX_AVC_SLICE_STATE

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

This is a slice level command and can be issued multiple times within a picture that is comprised of multiple slices. The same command is used for AVC encoder (PAK mode) and decoder (VLD and IT modes).

Programming Notes
MFX_AVC_SLICE_STATE command is not issued for AVC DXVA2 Short Format Bitstream decode, instead MFD_AVC_SLICEADDR command is executed to retrieve the next slice MB Start Address X and Y by H/W itself.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_AVC_SLICE_STATE Format: OpCode
	26:24	**Command Opcode** Default Value: 1h AVC Format: OpCode
	23:21	**SubOpcodeA** Default Value: 0h MFX_AVC_SLICE_STATE Format: OpCode
	20:16	**Command SubOpcodeB** Default Value: 3h MFX_AVC_SLICE_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 8h DWORD_COUNT_n Format: =n Excludes DWords 0,1
1	31:4	**Reserved** Format: MBZ

MFX_AVC_SLICE_STATE

	3:0	**Slice Type** It is set to the value of the syntax element read from the Slice Header. <table><tr><th>Value</th><th>Name</th></tr><tr><td>0000b</td><td>P Slice</td></tr><tr><td>0001b</td><td>B Slice</td></tr><tr><td>0010b</td><td>I Slice</td></tr><tr><td>0011b-1111b</td><td>Reserved</td></tr></table><table><tr><th>Programming Notes</th></tr><tr><td>Bits[3:2] must be 0</td></tr></table>
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. <table><tr><th>Value</th><th>Name</th></tr><tr><td>0-32</td><td></td></tr></table>
	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. <table><tr><th>Value</th><th>Name</th></tr><tr><td>0-32</td><td></td></tr></table>
	15:11	**Reserved**
	10:8	**Log 2 Weight Denom Chroma** Format: U3 <table><tr><th>Value</th><th>Name</th></tr><tr><td>0-7</td><td></td></tr></table>
	7:3	**Reserved**

<table>
<tr><th colspan="3">MFX_AVC_SLICE_STATE</th></tr>
<tr><td></td><td>2:0</td><td>Log 2 Weight Denom Luma
<table><tr><td>Format:</td><td>U3</td></tr></table>
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().
<table><tr><th>Value</th><th>Name</th></tr><tr><td>0-7</td><td></td></tr></table>
</td></tr>
<tr><td>3</td><td>31:30</td><td>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
<table><tr><th>Programming Notes</th></tr>
<tr><td>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.</td></tr>
<tr><td>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.</td></tr>
<tr><td>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.</td></tr></table>
</td></tr>
<tr><td></td><td>29</td><td>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.
<table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>Temporal</td></tr><tr><td>1</td><td>Spatial</td></tr></table>
</td></tr>
<tr><td></td><td>28:27</td><td>Disable Deblocking Filter Indicator
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>00b</td><td></td><td>FilterInternalEdgesFlag is set equal to 1</td></tr>
<tr><td>01b</td><td></td><td>Disable all deblocking operation, no deblocking parameter syntax element is read; filterInternalEdgesFlag is set equal to 0</td></tr>
<tr><td>10b</td><td></td><td>Macroblocks in different slices are considered not available;</td></tr></table>
</td></tr>
</table>

MFX_AVC_SLICE_STATE

		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: 0-2 | Name: **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.
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

MFX_AVC_SLICE_STATE

DWord	Bit	Description
		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, 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 0: Disable Value 1: Enable

MFX_AVC_SLICE_STATE

30 — ResetRateControlCounter

To reset the bit allocation accumulation counter to 0 to restart the rate control.

Value	Name
0	Not Reset
1	Reset

29:28 — RC Triggle Mode

Value	Name	Description
00b	Always Rate Control	Whereas RC becomes active if sum_act > sum_target or sum_act < sum_target
01b	Gentle Rate Control	whereas RC becomes active if sum_act > upper_midpt or sum_act < lower_midpt
10b	Loose Rate Control	whereas RC becomes active if sum_act > sum_max or sum_act < sum_min
11b	Reserved	

27:24 — RC Stable Tolerance

Format:	U4

This field specifies the tolerance required to deactivate RC once it has been triggered.

Value	Name
0-15	

23 — RC Panic Enable

If this field is set to 1, RC enters panic mode when sum_act > sum_max. RC Panic Type field controls what type of panic behavior is invoked.

Value	Name
0	Disable
1	Enable

22 — RC Panic Type

This field selects between two RC Panic methods

Value	Name
0	QP Panic
1	CBP Panic

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

21 — MB Type Direct Conversion Disable

Exists If:	//B-Slice

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

MFX_AVC_SLICE_STATE

		(continued)

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 the current slice encoded bits.

Programming Notes
Note: In VDEnc mode, the slice header PAK object maximum size is 25 DWs.

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

<table>
<tr><th colspan="3">MFX_AVC_SLICE_STATE</th></tr>
<tr><td></td><td></td><td><table><tr><td></td><td></td><td>encoded bits</td></tr><tr><td>1</td><td></td><td>Tail insertion into the output bitstream buffer is present, and is after the current slice encoded bits.</td></tr></table></td></tr>
<tr><td></td><td>14</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>13</td><td>EmulationByteSliceInsertEnable
To have PAK outputting SODB or EBSP to the output bitstream buffer<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td></td><td>outputting RBSP</td></tr><tr><td>1</td><td></td><td>outputting EBSP</td></tr></table></td></tr>
<tr><td></td><td>12</td><td>CabacZeroWordInsertionEnable
To pad the end of a SliceLayer RBSP to meet the encoded size requirement.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td></td><td>No Cabac_Zero_Word Insertion</td></tr><tr><td>1</td><td></td><td>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.</td></tr></table></td></tr>
<tr><td></td><td>11:8</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:4</td><td>Slice ID [3:0]
To identify the output data (coding information record) returned for rate control from PAK to ENC and VPP.</td></tr>
<tr><td></td><td>3:2</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>1:0</td><td>Stream ID [1:0]
To identify the output data (coding information record) returned for rate control from PAK to ENC and VPP.</td></tr>
<tr><td>7

Encoder Only</td><td>31:29</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>28:0</td><td>Indirect PAK-BSE Data Start Address (Write)<table><tr><td>Exists If:</td><td>//AVC Encode Mode</td></tr></table>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.</td></tr>
</table>

MFX_AVC_SLICE_STATE

<table>
<tr><td></td><td></td><td><table><tr><th>Value</th><th>Name</th></tr><tr><td>0 - 512MB</td><td></td></tr></table></td></tr>
<tr><td rowspan="6">8

Encoder Only</td><td>31:24</td><td>Magnitude of QP Max Negative Modifier<table><tr><td>Format:</td><td>U8</td></tr></table>This field specifies the lower limit of the QP modifier.<table><tr><th>Value</th><th>Name</th></tr><tr><td>0-51</td><td></td></tr></table></td></tr>
<tr><td>23:16</td><td>Magnitude of QP Max Positive Modifier<table><tr><td>Format:</td><td>U8</td></tr></table>This field specifies the upper limit of the QP modifier.<table><tr><th>Value</th><th>Name</th></tr><tr><td>0 - 15</td><td></td></tr></table></td></tr>
<tr><td>15:12</td><td>Shrink Param - Shrink Resistance<table><tr><td>Format:</td><td>U4</td></tr></table>This field specifies the additional points added each time decreased correction is invoked.<table><tr><th>Value</th><th>Name</th></tr><tr><td>0 - 15</td><td></td></tr></table></td></tr>
<tr><td>11:8</td><td>Shrink Param - Shrink Init<table><tr><td>Format:</td><td>U4</td></tr></table>This field specifies the initial points required to trip decreased control.<table><tr><th>Value</th><th>Name</th></tr><tr><td>0 - 15</td><td></td></tr></table></td></tr>
<tr><td>7:4</td><td>Grow Param - Grow Resistance<table><tr><td>Format:</td><td>U4</td></tr></table>This field specifies the additional points added each time increased correction is invoked.<table><tr><th>Value</th><th>Name</th></tr><tr><td>0 - 15</td><td></td></tr></table></td></tr>
<tr><td>3:0</td><td>Grow Param - Grow Init<table><tr><td>Format:</td><td>U4</td></tr></table>This field specifies the initial points required to trip increased control.<table><tr><th>Value</th><th>Name</th></tr><tr><td>0 - 15</td><td></td></tr></table></td></tr>
<tr><td>9

Encoder Only</td><td>31</td><td>RoundInterEnable<table><tr><td>Format:</td><td>Enable</td></tr></table>When this bit is not set, RoundInter defaults to 2.</td></tr>
</table>

MFX_AVC_SLICE_STATE

	30:28	**RoundInter** Format: U3 Rounding precision for Inter quantized coefficients 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.
	26:24	**RoundIntra** Format: U3 Rounding precision for Intra quantized coefficients 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** 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.

RoundInter values:

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

RoundIntra values:

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

Correct 6 values:

Value	Name
0 - 15	

MFX_AVC_SLICE_STATE

		<table><tr><th>Value</th><th>Name</th></tr><tr><td>0 - 15</td><td></td></tr></table>
	15:12	**Correct 4** <table><tr><td>Format:</td><td>U4</td></tr></table>This field specifies the points used in the fourth RC region when sum_act > lower_midpt but <= sum_target.<table><tr><th>Value</th><th>Name</th></tr><tr><td>0 - 15</td><td></td></tr></table>
	11:8	**Correct 3** <table><tr><td>Format:</td><td>U4</td></tr></table>This field specifies the points used in the third RC region when sum_act > sum_target but <= upper_midpt.<table><tr><th>Value</th><th>Name</th></tr><tr><td>0 - 15</td><td></td></tr></table>
	7:4	**Correct 2** <table><tr><td>Format:</td><td>U4</td></tr></table>This field specifies the points used in the second RC region when sum_act > upper_midpt but <= sum_max.<table><tr><th>Value</th><th>Name</th></tr><tr><td>0 - 15</td><td></td></tr></table>
	3:0	**Correct 1** <table><tr><td>Format:</td><td>U4</td></tr></table>This field specifies the points used in the topmost RC region when sum_act > sum_max.<table><tr><th>Value</th><th>Name</th></tr><tr><td>0 - 15</td><td></td></tr></table>
10 Encoder Only	31:28	**ClampValues - CV7**
	27:24	**CV6**
	23:20	**CV5**
	19:16	**CV4**
	15:12	**CV3**
	11:8	**CV2**
	7:4	**CV1**
	3:0	**CV0 - Clamp Value 0** <table><tr><td>Format:</td><td>U4</td></tr></table>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).

MFX_AVC_SLICE_STATE

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

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

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

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

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

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

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

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

Value	Name
0 - 15	

MFX_AVC_WEIGHTOFFSET_STATE

MFX_AVC_WEIGHTOFFSET_STATE

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_ AVC_ WEIGHTOFFSET_STATE Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h AVC_COMMON Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 5h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 60h Excludes DWord (0,1) Format: =n Total Length - 2

<table>
<tr><th colspan="3">MFX_AVC_WEIGHTOFFSET_STATE</th></tr>
<tr><td rowspan="2">1</td><td>31:1</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>0</td><td>Weight and Offset Select
It must be set in consistent with the WeightedPredFlag and WeightedBiPredIdc in the Img_State command. This parameter is specified for Intel interface only, not present in the DXVA. For implicit even though only one entry may be used, still loading the whole 32-entry table.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td>Weight and Offset L0 table</td><td>The list that followed is associated with the weight and offset for RefPicList L0</td></tr><tr><td>1</td><td>Weight and Offset L1 table</td><td>The list that followed is associated with the weight and offset for RefPicList L1</td></tr></table></td></tr>
<tr><td>2..97</td><td>31:0</td><td>WeightOffset<table><tr><td>WeightOffset[L=L0=0 or L1=1][i=0 to 31][Y=0/Cb=1/Cr=2][weight=0/offset=1] WeightOffset[L][i=0][Y=0][Weight=0], WeightOffset[L][i=0][Y=0][Offset=1] WeightOffset[L][i=0][Cb=1][Weight=0], WeightOffset[L][i=0][Cb=1][Offset=1] WeightOffset[L][i=0][Cr=2][Weight=0], WeightOffset[L][i=0][Cr=2][Offset=1]: WeightOffset[L][i=31][Y=0][Weight=0], WeightOffset[L][i=31][Y=0][Offset=1] WeightOffset[L][i=31][Cb=1][Weight=0], WeightOffset[L][i=31][Cb=1][Offset=1] WeightOffset[L][i=31][Cr=2][Weight=0], WeightOffset[L][i=31][Cr=2][Offset=1]</td></tr><tr><td>Format for explicit: Both Weight and Offset are S15 in two's compliment, with a valid range from -128 to 128 Format for implicit: S15</td></tr></table></td></tr>
</table>

MFX_AVC_WEIGHTOFFSET_STATE		
		This set of fields is always present whenever this command is issued. The full table, one entry for each reference picture, is always specified. Any reference list L0/L1[i] that does not exist, the corresponding weight and offset are set to 0. Weight and Offset are 2 byte each. Apair of Weight and Offset forms a dword, with Weight in the LOWER word and Offset in the HIGHER word. WeightOffset[L0=0][i=0 to 31][Y=0] (i.e. luma_weight_l0[i]) are specified for the weighting and offset factors applied to the luma prediction value for list 0 prediction using RefPicList0[i] (one-to-one correspondence in i). When luma_weight_l0_flag (Slice Header syntax element) is equal to 1, the value of luma_weight_l0[i] shall be in the range of -128 to 127. When luma_weight_l0_flag is equal to 0, luma_weight_l0[i] shall be inferred to be equal to 2luma_log2_weight_denom for RefPicList0[i]. luma_log2_weight_denom is a Slice Header syntax element. WeightOffset[L0=0][i=0 to 31][Cb=1] (i.e. chromaCb_weight_l0[i]) are specified for the weighting and offset factors applied to the chroma Cb prediction values for list 0 prediction using RefPicList0[i] (one-to-one correspondence in i). When chroma_weight_l0_flag (Slice Header syntax element) is equal to 1, the value of chromaCb_weight_l0[i] shall be in the range of -128 to 127. When chroma_weight_l0_flag is equal to 0, chromaCb_weight_l0[i] shall be inferred to be equal to 2chroma_log2_weight_denom for RefPicList0[i]. chroma_log2_weight_denom is a Slice Header syntax element. WeightOffset[L0=0][i=0 to 31][Cr=2] (i.e. chromaCr_weight_l0[i]) are specified for the weighting and offset factors applied to the chroma Cr prediction values for list 0 prediction using RefPicList0[i] (one-to-one correspondence in i). When chroma_weight_l0_flag (Slice Header syntax element) is equal to 1, the value of chromaCr_weight_l0[i] shall be in the range of -128 to 127. When chroma_weight_l0_flag is equal to 0, chromaCr_weight_l0[i] shall be inferred to be equal to 2chroma_log2_weight_denom for RefPicList0[i].

MFX_BSP_BUF_BASE_ADDR_STATE

MFX_BSP_BUF_BASE_ADDR_STATE

Project:	CHV, BSW
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** Project: All Format: MBZ
	11:0	**DWord Length** Default Value: 8h Excludes DWord (0,1)

MFX_BSP_BUF_BASE_ADDR_STATE

		Project: All Format: =n Total Length - 2
1	31:6	**BSD/MPC Row Store Scratch Buffer Base Address - Read/Write** 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** Project: CHV, BSW Format: MBZ
2 **Project:** CHV, BSW	31:16	**Reserved** Project: CHV, BSW Format: MBZ Reserved for 64-bit address extension.
	15:0	**BSD/MPC Row Store Scratch Buffer Base Address - Read/Write [47:32]** Project: CHV, BSW **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 **Project:**	31:15	**Reserved**
	14:13	**Reserved**

MFX_BSP_BUF_BASE_ADDR_STATE

CHV, BSW	12	**BSD/MPC Row Store Scratch Buffer Cache Select** Project: CHV, BSW 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** Project: CHV, BSW Format: MBZ
	10:9	**Reserved** Project: CHV, BSW Format: MBZ
	8:7	**BSD/MPC Row Store Scratch Buffer - Arbitration Priority Control** Project: CHV, BSW 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
	6:5	**Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for BSD/MPC Row Store Scratch Buffer Base Address** Project: CHV, BSW This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream. Value | Name 00b | Use Cacheability Controls from page table 01b | Uncacheable (UC) - non-cacheable 10b | Writethrough (WT) 11b | Writeback (WB)
	4:3	**Target Cache (TC) for BSD/MPC Row Store Scratch Buffer Base Address** Project: CHV, BSW This field allows the choice of LLC vs eLLC for caching Value | Name 00b | eLLC Only - not snooped in GT

MFX_BSP_BUF_BASE_ADDR_STATE

		01b — LLC Only 10b — LLC/eLLC Allowed 11b — L3, LLC, eLLC Allowed
	2	**Encrypted Data BSD/MPC Row Store Scratch Buffer Base Address** Project: CHV, BSW Format: Enable This field controls whether data is decrypted while being read. This field is ignored for writes.
	1:0	**Age for QUADLRU (AGE) BSD/MPC Row Store Scratch Buffer Base Address** Project: CHV, BSW Format: Enable This field allows the selection of AGE parameter for a given surface in LLC or eLLC.

Value	Name
11b	Good chance of generating hits.
10b	Next good chance of generating hits
01b	Decent chance of generating hits
00b	Poor chance of generating hits

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
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** Project: CHV, BSW Format: MBZ

MFX_BSP_BUF_BASE_ADDR_STATE

DWord	Bit	Description
5 **Project:** CHV, BSW	31:16	**Reserved** Project: All Format: MBZ Reserved for 64-bit address extension.
	15:0	**MPR Row Store Scratch Buffer Base Address - Read/Write [47:32]** Project: All This field is for the upper range of MPR Row Store Scratch Buffer Base Address. This field is used for 48-bit addressing.
6 **Project:** CHV, BSW	31:15	**Reserved** Project: CHV, BSW Format: MBZ
	14:13	**Reserved** Project: CHV, BSW Format: MBZ
	12	**MPR Row Store Scratch Buffer Cache Select** Project: CHV, BSW 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** Project: CHV, BSW Format: MBZ
	10:9	**Reserved** Project: CHV, BSW Format: MBZ
	8:7	**MPR Row Store Scratch Buffer - Arbitration Priority Control** Project: CHV, BSW 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

MFX_BSP_BUF_BASE_ADDR_STATE

	6:5	**Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for MPR Row Store Scratch Buffer Base Address** Project: CHV, BSW This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream. **Value** / **Name** 00b: Use Cacheability Controls from page table 01b: Uncacheable (UC) - non-cacheable 10b: Writethrough (WT) 11b: Writeback (WB)
	4:3	**Target Cache (TC) MPR Row Store Scratch Buffer Base Address** Project: CHV, BSW This field allows the choice of LLC vs eLLC for caching **Value** / **Name** 00b: eLLC Only - not snooped in GT 01b: LLC Only 10b: LLC/eLLC Allowed 11b: L3, LLC, eLLC Allowed
	2	**Encrypted Data MPR Row Store Scratch Buffer Base Address** Project: CHV, BSW Format: Enable This field controls whether data is decrypted while being read. This field is ignored for writes.
	1:0	**Age for QUADLRU (AGE) MPR Row Store Scratch Buffer Base Address** Project: CHV, BSW Format: Enable This field allows the selection of AGE parameter for a given surface in LLC or eLLC. **Value** / **Name** 11b: Good chance of generating hits. 10b: Next good chance of generating hits 01b: Decent chance of generating hits 00b: Poor chance of generating hits
7	31:6	**Bitplane Read Buffer Base Address** Project: All It must be cacheline aligned (i.e. 64 bytes address boundary), so lower bit 0 to 5 are used for controlling information. 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

<table>
<tr><th colspan="3">MFX_BSP_BUF_BASE_ADDR_STATE</th></tr>
<tr><td rowspan="2"></td><td></td><td>: 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.</td></tr>
<tr><td>5:0</td><td>Reserved
Project: CHV, BSW
Format: MBZ</td></tr>
<tr><td rowspan="2">8
Project:
CHV, BSW</td><td>31:16</td><td>Reserved
Project: CHV, BSW
Format: MBZ
Reserved for 64-bit address extension.</td></tr>
<tr><td>15:0</td><td>Bitplane Read Buffer Base Address - Read/Write [47:32]
Project: All
This field is for the upper range of Bitplane Read Buffer Base Address. This field is used for 48-bit addressing.</td></tr>
<tr><td rowspan="5">9
Project:
CHV, BSW</td><td>31:15</td><td>Reserved
Project: CHV, BSW
Format: MBZ</td></tr>
<tr><td>14:13</td><td>Reserved
Project: CHV, BSW
Format: MBZ</td></tr>
<tr><td>12:11</td><td>Reserved
Project: CHV, BSW
Format: MBZ</td></tr>
<tr><td>10:9</td><td>Reserved
Project: CHV, BSW
Format: MBZ</td></tr>
<tr><td>8:7</td><td>Bitplane Read Buffer - Arbitration Priority Control
Project: CHV, BSW
Format: U2
This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.
<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>Highest priority</td></tr><tr><td>01b</td><td>Second highest priority</td></tr><tr><td>10b</td><td>Third highest priority</td></tr><tr><td>11b</td><td>Lowest priority</td></tr></table></td></tr>
</table>

MFX_BSP_BUF_BASE_ADDR_STATE

<table>
<tr><td></td><td>6:5</td><td>Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for Bitplane Read Buffer Base Address
Project: CHV, BSW
This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>00b</td><td>Use Cacheability Controls from page table</td></tr>
<tr><td>01b</td><td>Uncacheable (UC) - non-cacheable</td></tr>
<tr><td>10b</td><td>Writethrough (WT)</td></tr>
<tr><td>11b</td><td>Writeback (WB)</td></tr>
</table></td></tr>
<tr><td></td><td>4:3</td><td>Target Cache (TC) Bitplane Read Buffer Base Address
Project: CHV, BSW
This field controls the L3$, LLC and eLLC (eDRAM) cacheability for a given surface. Setting of "00" points to PTE settings which defaults to eDRAM (when present). If no eDRAM, the access will be allocated to LLC. Setting of "01", allocates into LLC and victimizes the line to eDRAM. Setting of "10" allows the line to be allocated in either LLC or eDRAM. Setting of "11" is the only option for a memory access to be allocated in L3$ as well as LLC/eLLC
00b: eLLC Only ("00" setting points TC selection to PTE which defaults to eLLC)
01b: LLC Only (Works at the allocation time, later victimization from LLC downgrades the line to eLLC if present).
10b: LLC/eLLC Allowed.
11b: L3, LLC, eLLC Allowed.
Errata CHV:A-E (FIXED BY:G0 Stepping):
For all system that does NOT use SVM (i.e. coherent L3$ surfaces), back snoops from LLC has to be disabled (Dis_GtCvUpdtOnRd = “1”). Than target Cache settings can be programmed as POR requirements of L3/LLC/eDRAM caching.
For all systems that does use SVM (i.e. coherent L3$ surfaces), the recomended setting would be "00" in target cache settings. In case of L3 surfaces, the performance has to be tuned between "00" and "11" setting based on the benefits of L3 caching outweighting the degradation of backsnoops.
Post G0-stepping, the above w/a for coherent L3$ surfaces is not needed.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>00b</td><td>eLLC Only - not snooped in GT</td></tr>
<tr><td>01b</td><td>LLC Only</td></tr>
<tr><td>10b</td><td>LLC/eLLC Allowed</td></tr>
<tr><td>11b</td><td>L3, LLC, eLLC Allowed</td></tr>
</table></td></tr>
<tr><td></td><td>2</td><td>Encrypted Data Bitplane Read Buffer Base Address
Project: CHV, BSW
Format: Enable
This field controls whether data is decrypted while being read. This field is ignored for writes.</td></tr>
</table>

MFX_BSP_BUF_BASE_ADDR_STATE		
	1:0	**Age for QUADLRU (AGE) Bitplane Read Buffer Base Address** Project: CHV, BSW Format: Enable This field allows the selection of AGE parameter for a given surface in LLC or eLLC. If a particular allocation is done at youngest age ("0,1,2") it tends to stay longer in the cache. This option is given to GFX software to be able to decide which surfaces are more likely to generate HITs, hence need to be replaced least often in caches.

Value	Name
11b	Good chance of generating hits
10b	Next good chance of generating hits
01b	Decent chance of generating hits
00b	Poor chance of generating hits

MFX_DBK_OBJECT

MFX_DBK_OBJECT

Project:	CHV, BSW
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 MFX_DBK_OBJECT Format: OpCode
	26:24	**Media Command Opcode** Default Value: 0h Common Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 9h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 0Bh Excludes DWord (0,1) Format: =n Note: Regardless of the mode, inline data must be present in this command
1	31:6	**Pre Deblocking Source Address** Format: GraphicsAddress[31:6] Specifies the 4K byte aligned frame buffer address for outputting the non-filtered reconstructed YUV picture (i.e. output of final adder in each codec standard, and prior to the deblocking filter unit).
	5:0	**Reserved** Project: CHV, BSW Format: MBZ

MFX_DBK_OBJECT

DWord	Bit	Description
2 **Project:** CHV, BSW	31:16	**Reserved** Project: CHV, BSW Format: MBZ
	15:0	**Pre Deblocking Source Address High** Project: CHV, BSW Format: GraphicsAddress[47:32] This field is for the upper range of Pre-Deblocking Source Address. This field is used for 48-bit addressing.
3 **Project:** CHV, BSW	31:15	**Reserved**
	14:13	**Reserved**
	12:11	**Reserved** Project: CHV, BSW Format: MBZ
	10:9	**Reserved** Project: CHV, BSW Format: MBZ
	8:7	**Pre Deblocking Source - Arbitration Priority Control** Project: CHV, BSW 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
	6:5	**Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for Pre Deblocking Source Address** Project: CHV, BSW This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream. **Value** / **Name**: 00b: Use Cacheability Controls from page table 01b: Uncacheable (UC) - non-cacheable 10b: Writethrough (WT) 11b: Writeback (WB)

MFX_DBK_OBJECT

DWord	Bit	Description
	4:3	**Target Cache (TC) for Pre Deblocking Source Address** Project: CHV, BSW This field allows the choice of LLC vs eLLC for caching **Value / Name** 00b: eLLC Only - not snooped in GT 01b: LLC Only 10b: LLC/eLLC Allowed 11b: L3, LLC, eLLC Allowed
	2	**Encrypted Data for Pre Deblocking Source Address** Project: CHV, BSW Format: Enable This field controls whether data is decrypted while being read. This field is ignored for writes.
	1:0	**Age for QUADLRU (AGE) for Pre Deblocking Source Address** Project: CHV, BSW Format: Enable This field allows the selection of AGE parameter for a given surface in LLC or eLLC. **Value / Name** 11b: Good chance of generating hits. 10b: Next good chance of generating hits 01b: Decent chance of generating hits 00b: Poor chance of generating hits
4	31:6	**Deblocking Control Address** Format: GraphicsAddress[31:6] Specifies the 4K byte aligned frame buffer address as input MB-level deblocking parameters to control the way hardware deblock the each micro-block. One 512-bit cacheline is allocated for each Macroblock in raster scan order.
	5:0	**Reserved** Project: CHV, BSW Format: MBZ
5 **Project:** CHV, BSW	31:16	**Reserved** Project: CHV, BSW Format: MBZ
	15:0	**Deblocking Control Address High** Project: CHV, BSW Format: GraphicsAddress[47:32] This field is for the upper range of Deblocking Control Address (DeblockCntrlAddr). This field is used for 48-bit addressing.

MFX_DBK_OBJECT

DWord	Bit	Description
6 **Project:** CHV, BSW	31:15	**Reserved** Format: MBZ
	14:13	**Reserved** Project: CHV, BSW Format: MBZ
	12:11	**Reserved** Project: CHV, BSW Format: MBZ
	10:9	**Reserved** Project: CHV, BSW Format: MBZ
	8:7	**Deblocking Control - Arbitration Priority Control** Project: CHV, BSW 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
	6:5	**Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for Deblocking Control Address** Project: CHV, BSW This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream. Value / Name: 00b: Use Cacheability Controls from page table 01b: Uncacheable (UC) - non-cacheable 10b: Writethrough (WT) 11b: Writeback (WB)
	4:3	**Target Cache (TC) for Deblocking Control Address** Project: CHV, BSW This field allows the choice of LLC vs eLLC for caching Value / Name: 00b: eLLC Only - not snooped in GT 01b: LLC Only 10b: LLC/eLLC Allowed 11b: L3, LLC, eLLC Allowed

<table>
<tr><th colspan="3">MFX_DBK_OBJECT</th></tr>
<tr><td rowspan="2"></td><td>2</td><td>Encrypted Data for Deblocking Control Address
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>Enable</td></tr></table>
This field controls whether data is decrypted while being read. This field is ignored for writes.</td></tr>
<tr><td>1:0</td><td>Age for QUADLRU (AGE) for Deblocking Control Address
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>Enable</td></tr></table>
This field allows the selection of AGE parameter for a given surface in LLC or eLLC.
<table><tr><th>Value</th><th>Name</th></tr><tr><td>11b</td><td>Good chance of generating hits.</td></tr><tr><td>10b</td><td>Next good chance of generating hits</td></tr><tr><td>01b</td><td>Decent chance of generating hits</td></tr><tr><td>00b</td><td>Poor chance of generating hits</td></tr></table></td></tr>
<tr><td rowspan="2">7</td><td>31:6</td><td>Deblocking Destination Address
<table><tr><td>Format:</td><td>GraphicsAddress[31:6]</td></tr></table>
Specifies the 4K byte aligned frame buffer address for outputting the post-loop filtered reconstructed YUV picture (i.e. output of the deblocking filter unit)</td></tr>
<tr><td>5:0</td><td>Reserved</td></tr>
<tr><td rowspan="2">8
Project:
CHV, BSW</td><td>31:16</td><td>Reserved</td></tr>
<tr><td>15:0</td><td>Deblocking Destination Address High
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>GraphicsAddress[47:32]</td></tr></table>
This field is for the upper range of Deblocking Destination Address. This field is used for 48-bit addressing.</td></tr>
<tr><td rowspan="5">9
Project:
CHV, BSW</td><td>31:15</td><td>Reserved</td></tr>
<tr><td>14:13</td><td>Reserved</td></tr>
<tr><td>12:11</td><td>Reserved</td></tr>
<tr><td>10:9</td><td>Reserved</td></tr>
<tr><td>8:7</td><td>Deblocking Destination - Arbitration Priority Control
<table><tr><td>Project:</td><td>CHV, BSW</td></tr></table>
This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.
<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>Highest priority</td></tr><tr><td>01b</td><td>Second highest priority</td></tr><tr><td>10b</td><td>Third highest priority</td></tr><tr><td>11b</td><td>Lowest priority</td></tr></table></td></tr>
</table>

MFX_DBK_OBJECT

DWord	Bit	Description
	6:5	**Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for Deblocking Destination Address** Project: CHV, BSW This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream. **Value** / **Name** 00b: Use Cacheability Controls from page table 01b: Uncacheable (UC) - non-cacheable 10b: Writethrough (WT) 11b: Writeback (WB)
	4:3	**Target Cache (TC) for Deblocking Destination Address** Project: CHV, BSW This field allows the choice of LLC vs eLLC for caching **Value** / **Name** 00b: eLLC Only - not snooped in GT 01b: LLC Only 10b: LLC/eLLC Allowed 11b: L3, LLC, eLLC Allowed
	2	**Encrypted Data for Deblocking Destination Address** Project: CHV, BSW Format: Enable This field controls whether data is decrypted while being read. This field is ignored for writes.
	1:0	**Age for QUADLRU (AGE) for Deblocking Destination Address** Project: CHV, BSW Format: Enable This field allows the selection of AGE parameter for a given surface in LLC or eLLC. **Value** / **Name** 11b: Good chance of generating hits 10b: Next good chance of generating hits 01b: Decent chance of generating hits 00b: Poor chance of generating hits
10	31:6	**Deblock Row Store Address** Format: GraphicsAddress[31:6] This field provides the base address of the scratch buffer (read and write) used by the deblocking filter unit to store MB information of the previous row for filtering of each macroblock in the current row. The Deblocking Filter Row Store buffer must be 64-byte

MFX_DBK_OBJECT

<table>
<tr><td></td><td></td><td>cacheline aligned.Hardware uses the horizontal address of the current macroblock to address the Deblocking Filter Row Store.
Programming Notes
This is one of the four RowStore Scratch Buffers which can programmed to use the internal Media Cache (total size 640 CacheLine). When Deblocking Row Store 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 cachelines address location for this buffer. Driver needs to make sure the whole buffer fits into Media Internal Storage. Also, this command is only for Standalone Deblocker Mode. During Standalone Deblocker Mode, only this Deblock Row Store will be active. There is no other row store active so it can use the whole internal Media Cache. (Notes: 2 cachelines per MB for non-mbaff; 4 cachelines per MB pair for mbaff, and the buffer needs to have enough space for 1 MB (pair) row).</td></tr>
<tr><td></td><td>5:0</td><td>Reserved
Project: CHV, BSW
Format: MBZ</td></tr>
<tr><td>11
Project:
CHV, BSW</td><td>31:16</td><td>Reserved
Project: CHV, BSW
Format: MBZ
Reserved for 64-bit address extension.</td></tr>
<tr><td></td><td>15:0</td><td>Deblock Row Store Address High
Project: CHV, BSW
Format: GraphicsAddress[47:32]
This field is for the upper range of Deblock Row Store Address (DeblockRowStoreAddr). This field is used for 48-bit addressing.</td></tr>
<tr><td>12
Project:
CHV, BSW</td><td>31:15</td><td>Reserved
Project: CHV, BSW
Format: MBZ</td></tr>
<tr><td></td><td>14:13</td><td>Reserved
Project: CHV, BSW
Format: MBZ</td></tr>
<tr><td></td><td>12</td><td>Deblock Row Store - Cache Select
Project: CHV, BSW
This field controls if Deblock Row Store is going to store inside Media Cache or to LLC.
Value | Name
0 | Buffer going to LLC.
1 | Buffer going to Internal Media Storage</td></tr>
</table>

MFX_DBK_OBJECT

	Bits	Description
	11	**Reserved** Project: CHV, BSW Format: MBZ
	10:9	**Reserved** Project: CHV, BSW Format: MBZ
	8:7	**Deblock Row Store - Arbitration Priority Control** Project: CHV, BSW 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
	6:5	**Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for Deblock Row Store Address** Project: CHV, BSW This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream. Value / Name: 00b: Use Cacheability Controls from page table 01b: Uncacheable (UC) - non-cacheable 10b: Writethrough (WT) 11b: Writeback (WB)
	4:3	**Target Cache (TC) for Deblock Row Store Address** Project: CHV, BSW This field controls the L3$, LLC and eLLC (eDRAM) cacheability for a given surface. Setting of "00" points to PTE settings which defaults to eDRAM (when present). If no eDRAM, the access will be allocated to LLC. Setting of "01", allocates into LLC and victimizes the line to eDRAM. Setting of "10" allows the line to be allocated in either LLC or eDRAM. Setting of "11" is the only option for a memory access to be allocated in L3$ as well as LLC/eLLC 00b: eLLC Only *("00" setting points TC selection to PTE which defaults to eLLC)* 01b: LLC Only (*Works at the allocation time, later victimization from LLC downgrades the line to eLLC if present*). 10b: LLC/eLLC Allowed. 11b: L3, LLC, eLLC Allowed. **Errata CHV:A-E (FIXED BY:G0 Stepping):** For all system that does NOT use SVM (i.e. coherent L3$ surfaces), back snoops from LLC has to be disabled (**Dis_GtCvUpdtOnRd = "1"**). Than target Cache settings can be programmed as POR requirements of L3/LLC/eDRAM caching.

MFX_DBK_OBJECT

		For all systems that does use SVM (i.e. coherent L3$ surfaces), the recomended setting would be "00" in target cache settings. In case of L3 surfaces, the performance has to be tuned between "00" and "11" setting based on the benefits of L3 caching outweighting the degradation of backsnoops. Post G0-stepping, the above w/a for coherent L3$ surfaces is not needed.

Value	Name
00b	eLLC Only - not snooped in GT
01b	LLC Only
10b	LLC/eLLC Allowed
11b	L3, LLC, eLLC Allowed

	2	**Encrypted Data for Deblock Row Store Address**

Project:	CHV, BSW
Format:	Enable

This field controls whether data is decrypted while being read. This field is ignored for writes.

	1:0	**Age for QUADLRU (AGE) for Deblock Row Store Address**

Project:	CHV, BSW
Format:	Enable

This field allows the selection of AGE parameter for a given surface in LLC or eLLC. If a particular allocation is done at youngest age (“0,1,2”) it tends to stay longer in the cache. This option is given to GFX software to be able to decide which surfaces are more likely to generate HITs, hence need to be replaced least often in caches.

Value	Name
11b	Good chance of generating hits
10b	Next good chance of generating hits
01b	Decent chance of generating hits
00b	Poor chance of generating hits

MFX_FQM_STATE

MFX_FQM_STATE

Project: CHV, BSW
Source: VideoCS
Length Bias: 2

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

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

MFX_FQM_STATE

	1:0	**AVC**

Exists If:	//AVC- Decoder Only

For AVC QM Type: This field specifies which Quantizer Matrix is loaded.

Value	Name
0	AVC_4x4_Intra_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs)
1	AVC_4x4_Inter_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs)
2	AVC_8x8_Intra_MATRIX
3	AVC_8x8_Inter_MATRIX

	1:0	**MPEG2**

Exists If:	//MPEG2- Decoder Only

For MPEG2 QM Type: This field specifies which Quantizer Matrix is loaded.

Value	Name
0	MPEG_INTRA_QUANTIZER_MATRIX
1	MPEG_NON_INTRA_QUANTIZER_MATRIX
2-3	Reserved

	1:0	**JPEG**

Project:	CHV, BSW
Exists If:	//JPEG- Encoder Only

For JPEG QM Type:This field specifies which Quantizer Matrix is loaded.

Value	Name
0	JPEG_Luma_Y_QUANTIZER_MATRIX (or R)
1	JPEG_Chroma_Cb_QUANTIZER_MATRIX (or G)
2	JPEG_Chroma_Cr_QUANTIZER_MATRIX (or B)

Programming Notes
For JPEG encoder, each quantization element presents 16-bit 1/QM[i][j]. In RGB encoding, because the order input image components can be RGB, GBR, BGR, YUV, the value 0 is used for the first image component, the value 1 is used for the second image component, and the value 2 is used for the third image component.

2..33	31:0	**Forward Quantizer Matrix**

Project:	All
Format:	U32

The format of a Quantizer Matrix is an 8x8 matrix in raster order. Each element is an unsigned byte.

MFX_IND_OBJ_BASE_ADDR_STATE

MFX_IND_OBJ_BASE_ADDR_STATE

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

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

The MFX_IND_OBJ_BASE_ADDR command sets the memory base address pointers for the corresponding Indirect Object Data Start Addresses (Offsets) specified in each OBJECT commands. The characteristic of these indirect object data is their variable size (per MB or per Slice). Hence, each OBJECT command must specify the indirect object data offset from the base address to start fetching or writing object data.

While the use of base address is unconditional, the indirection can be effectively disabled by setting the base address to zero.
For decoder, there are:

- 1 read-only per-slice indirect object in the BSD_OBJECT Command, and
- 2 read-only per-MB indirect objects in the IT_OBJECT Command.

For decoder: the Video Command Streamer (VCS) will perform the memory access bound check automatically using the corresponding MFC Indirect Object Access Upper Bound specification. If any access is at or beyond the upper bound, zero value is returned. The request to memory is still being sent, but the corresponding codec's BSD unit will detect this condition and perform the zeroing return. If the Upper Bound is turned off, the beyond bound request will return whatever on the bus (invalid data).
For encoder, there are:

- 1 read-only per-MB indirect object in the PAK_OBJECT Command, and
- 1 write-only per-slice indirect object in the PAK Slice_State Command

For encoder: whenever an out of bound address accessing request is generated, VMX will detect such requests and snap the address to the corresponding [indirect object base address + indirect data start address]. VMX will return all 0s as the data to the requestor. NotationDefinitionPhysicalAddress[n:m] Corresponding bits of a physical graphics memory byte address (not mapped by a GTT) GraphicsAddress[n:m] Corresponding bits of an absolute, virtual graphics memory byte address (mapped by a GTT).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode

MFX_IND_OBJ_BASE_ADDR_STATE

	28:27	**Pipeline** Default Value: 2h MFX_IND_OBJ_BASE_ADDR_STATE Format: OpCode
	26:24	**Common Opcode** Default Value: 0h MFX_IND_OBJ_BASE_ADDR_STATE Format: OpCode
	23:21	**Sub OpcodeA** Default Value: 0h MFX_IND_OBJ_BASE_ADDR_STATE Format: OpCode
	20:16	**SubOpcodeB** Default Value: 3h MFX_IND_OBJ_BASE_ADDR_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 0018h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:12	**MFX Indirect Bitstream Object - Base Address (Decoder and Stitch Modes)** Project: All Format: GraphicsAddress[31:12] Specifies the 4K-byte aligned memory base address for the read-only indirect data object pointed in the MFD_XXX_BSD_OBJECT command for fetching (reading) the compressed Slice Data.This field is only valid in MPEG2, AVC and VC1 decoder VLD mode.
	11:6	**Reserved** Format: MBZ
	5:0	**Reserved** Project: CHV, BSW Format: MBZ
2	31:16	**Reserved** Project: All Format: MBZ Reserved for 64-bit address extension.
	15:0	**MFX Indirect Bitstream Object - Destination Address (Decoder and Stitch Modes)[47:32]** Project: All

MFX_IND_OBJ_BASE_ADDR_STATE		
		Description This field is for the upper range of MFX Indirect Bitstream Object Base Address. This field is used for 48-bit addressing.
3	31:15	**Reserved** Project: CHV, BSW Format: MBZ
	14:13	**Reserved** Project: CHV, BSW Format: MBZ
	12:11	**Reserved** Project: CHV, BSW Format: MBZ
	10:9	**Reserved** Project: CHV, BSW Format: MBZ
	8:7	**MFX Indirect Bitstream ObjectBase - Arbitration Priority Control** Project: CHV, BSW 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
	6:5	**Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for MFX Indirect Bitstream ObjectBase Address** Project: CHV, BSW This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream. Value / Name: 00b: Use Cacheability Controls from page table 01b: Uncacheable (UC) - non-cacheable 10b: Writethrough (WT) 11b: Writeback (WB)
	4:3	**Target Cache (TC) MFX Indirect Bitstream ObjectBase Address** Project: CHV, BSW

MFX_IND_OBJ_BASE_ADDR_STATE

<table>
<tr><td></td><td></td><td>This field allows the choice of LLC vs eLLC for caching
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>00b</td><td>eLLC Only - not snooped in GT</td></tr>
<tr><td>01b</td><td>LLC Only</td></tr>
<tr><td>10b</td><td>LLC/eLLC Allowed</td></tr>
<tr><td>11b</td><td>L3, LLC, eLLC Allowed</td></tr>
</table></td></tr>
<tr><td></td><td>2</td><td>Encrypted Data MFX Indirect Bitstream ObjectBase Address
<table>
<tr><td>Project:</td><td>CHV, BSW</td></tr>
<tr><td>Format:</td><td>Enable</td></tr>
</table>
This field controls whether data is decrypted while being read. This field is ignored for writes.</td></tr>
<tr><td></td><td>1:0</td><td>Age for QUADLRU (AGE) MFX Indirect Bitstream ObjectBase Address
<table>
<tr><td>Project:</td><td>CHV, BSW</td></tr>
<tr><td>Format:</td><td>Enable</td></tr>
</table>
This field allows the selection of AGE parameter for a given surface in LLC or eLLC.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>11b</td><td>Good chance of generating hits.</td></tr>
<tr><td>10b</td><td>Next good chance of generating hits</td></tr>
<tr><td>01b</td><td>Decent chance of generating hits</td></tr>
<tr><td>00b</td><td>Poor chance of generating hits</td></tr>
</table></td></tr>
<tr><td>4</td><td>31:12</td><td>MFX Indirect Bitstream Object - Access Upper Bound (Decoder and Stitch Modes)
<table>
<tr><td>Format:</td><td>GraphicsAddress[31:12]</td></tr>
</table>
This field specifies the 4K-byte aligned (exclusive) maximum Graphics Memory address access by the indirect data object in the MFD_XXX_BSD_OBJECT command for the compressed Slice Data. Indirect data accessed at this address and beyond will return as 0 by the hardware. Setting this field to 0 will cause this range check to be ignored.If non-zero, this address must be greater than the MFX Indirect Bitstream ObjectBase Address state.Hardware ignores this field if indirect data is not present, i.e. the Indirect Data Length field of the MFD_XXX_BSD_OBJECT command is set to 0.This field is only valid in MPEG2, AVC, VP8, and VC1 decoder VLD mode.
<table>
<tr><th>Programming Notes</th><th>Project</th></tr>
<tr><td>For VP8 Encoder, this field is corresponding to MFC Indirect PAK-BSE Object - Access Upper Bound in DW24, DW25. Please program Indirect bitstream upperbound in this field the same as DW24, DW25.</td><td>CHV, BSW</td></tr>
</table></td></tr>
<tr><td></td><td>11:0</td><td>Reserved
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td>5
Project:
CHV,</td><td>31:16</td><td>Reserved
<table>
<tr><td>Project:</td><td>CHV, BSW</td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
</table>

MFX_IND_OBJ_BASE_ADDR_STATE		
BSW		Reserved for 64-bit address extension.
	15:0	**MFX Indirect Bitstream Object UpperBound (Decoder and Stitch Modes)[47:32]** Project: CHV, BSW **Description** / **Project**: This field is for the upper range of MFX Indirect Bitstream Object UpperBound. This field is used for 48-bit addressing. — CHV, BSW **Programming Notes** / **Project**: For **VP8 Encoder**, this field is corresponding to **MFC Indirect PAK-BSE Object - Access Upper Bound in DW24, DW25**. Please program Indirect bitstream upperbound in this field the same as DW24, DW25. — CHV, BSW
6	31:12	**MFX Indirect MV Object - Base Address** Format: GraphicsAddress[31:12] Specifies the 4K-byte aligned memory base address for the read-only indirect data object pointed in the encoder MFC_AVC_PAK_OBJECT command or the decoder MFD_IT_OBJECT command for fetching the per-MB MV data.This field is only valid in AVC encoder mode or in AVC decoder IT mode
	11:6	**Reserved** Format: MBZ
	5:0	**Reserved** Project: CHV, BSW Format: MBZ
7 **Project:** CHV, BSW	31:16	**Reserved** Project: CHV, BSW Format: MBZ Reserved for 64-bit address extension.
	15:0	**MFX Indirect MV Object Base Address [47:32]** Project: All **Description** / **Project**: This field is for the upper range of MFX Indirect MV Object Base Address. This field is used for 48-bit addressing. — CHV, BSW
8 **Project:** CHV,	31:15	**Reserved** Project: CHV, BSW Format: MBZ

MFX_IND_OBJ_BASE_ADDR_STATE

BSW	14:13	**Reserved** Project: CHV, BSW Format: MBZ
	12:11	**Reserved** Project: CHV, BSW Format: MBZ
	10:9	**Reserved** Project: CHV, BSW Format: MBZ
	8:7	**MFX Indirect MV Object - Arbitration Priority Control** Project: CHV, BSW 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
	6:5	**Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for MFX Indirect MV ObjectBase Address** Project: CHV, BSW This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream. **Value** / **Name** 00b: Use Cacheability Controls from page table 01b: Uncacheable (UC) - non-cacheable 10b: Writethrough (WT) 11b: Writeback (WB)
	4:3	**Target Cache (TC) MFX Indirect MV ObjectBase Address** Project: CHV, BSW This field allows the choice of LLC vs eLLC for caching **Value** / **Name** 00b: eLLC Only - not snooped in GT 01b: LLC Only 10b: LLC/eLLC Allowed 11b: L3, LLC, eLLC Allowed

MFX_IND_OBJ_BASE_ADDR_STATE

DWord	Bit	Description
	2	**Encrypted Data MFX Indirect MV ObjectBase Address** Project: CHV, BSW Format: Enable This field controls whether data is decrypted while being read. This field is ignored for writes.
	1:0	**Age for QUADLRU (AGE) MFX Indirect MV ObjectBase Address** Project: CHV, BSW Format: Enable This field allows the selection of AGE parameter for a given surface in LLC or eLLC. Value / Name: 11b: Good chance of generating hits. 10b: Next good chance of generating hits 01b: Decent chance of generating hits 00b: Poor chance of generating hits
9	31:12	**MFX Indirect MV Object Access Upper Bound** Format: GraphicsAddress[31:12] This field specifies the 4K-byte aligned (exclusive) maximum Graphics Memory address access by the indirect data object in the MFC_AVC_PAK_OBJECT / MFD_IT_OBJECT command for the per-MB MV data. Indirect data accessed at this address and beyond will return as 0 by the hardware. Setting this field to 0 will cause this range check to be ignored.If non-zero, this address must be greater than the MFX Indirect MV Object Base Address state.Hardware ignores this field if indirect data is not present, i.e. the Indirect Data Length field of the MFC_AVC_PAK_OBJECT / MFD_IT_OBJECT command is set to 0.This field is only valid in AVC encoder mode or in AVC decoder IT mode.
	11:0	**Reserved** Format: MBZ
10 **Project:** CHV, BSW	31:16	**Reserved** Project: All Format: MBZ Reserved for 64-bit address extension.
	15:0	**MFX Indirect MV Object UpperBound [47:32]** Project: All Description / Project: This field is for the upper range of MFX Indirect MV Object Base Address. — This field is used for 48-bit addressing. — CHV, BSW

MFX_IND_OBJ_BASE_ADDR_STATE

DWord	Bit	Description
11	31:12	**MFD Indirect IT-COEFF Object - Base Address (Decoder Only)** Format: GraphicsAddress[31:12] Specifies the 4K-byte aligned memory base address for the read-only indirect data object pointed in the MFD_IT_OBJECT command for fetching (reading) the per-MB non-scaled coefficient data (all inverse scaling and quantization are done in hardware).This field is only valid in MPEG2, AVC and VC1 decoder IT mode.
	11:6	**Reserved** Format: MBZ
	5:0	**Reserved** Project: CHV, BSW Format: MBZ
12 **Project:** CHV, BSW	31:16	**Reserved** Project: CHV, BSW Format: MBZ Reserved for 64-bit address extension.
	15:0	**MFD Indirect IT-COEFF Object Base Address [47:32]** Project: CHV, BSW **Description** / **Project** This field is for the upper range of MFX Indirect IT-COEFF Object Base Address. / This field is for the upper range of MFX Indirect MV Object Base Address. / This field is used for 48-bit addressing. / CHV, BSW
13 **Project:** CHV, BSW	31:15	**Reserved** Project: CHV, BSW Format: MBZ
	14:13	**Reserved** Project: CHV, BSW Format: MBZ
	12:11	**Reserved** Project: CHV, BSW Format: MBZ
	10:9	**Reserved**
	8:7	**MFD Indirect IT-COEFF Object Desitnation - Arbitration Priority Control** Project: CHV, BSW Format: U2 This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.

MFX_IND_OBJ_BASE_ADDR_STATE

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

6:5 **Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for MFD Indirect IT-COEFF ObjectBase Address**

Project:	CHV, BSW

This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream.

Value	Name
00b	Use Cacheability Controls from page table
01b	Uncacheable (UC) - non-cacheable
10b	Writethrough (WT)
11b	Writeback (WB)

4:3 **Target Cache (TC) MFD Indirect IT-COEFF ObjectBase Address**

Project:	CHV, BSW

This field allows the choice of LLC vs eLLC for caching

Value	Name
00b	eLLC Only - not snooped in GT
01b	LLC Only
10b	LLC/eLLC Allowed
11b	L3, LLC, eLLC Allowed

2 **Encrypted Data MFD Indirect IT-COEFF ObjectBase Address**

Project:	CHV, BSW
Format:	Enable

This field controls whether data is decrypted while being read. This field is ignored for writes.

1:0 **Age for QUADLRU (AGE) MFD Indirect IT-COEFF ObjectBase Address**

Project:	CHV, BSW
Format:	Enable

This field allows the selection of AGE parameter for a given surface in LLC or eLLC.

Value	Name
11b	Good chance of generating hits.
10b	Next good chance of generating hits
01b	Decent chance of generating hits
00b	Poor chance of generating hits

MFX_IND_OBJ_BASE_ADDR_STATE

DWord	Bit	Description
14	31:12	**MFD Indirect IT-COEFF Object - Access Upper Bound (Decoder Only)** Format: GraphicsAddress[31:12] This field specifies the 4K-byte aligned (exclusive) maximum Graphics Memory address access by the indirect data object in the MFD_IT_OBJECT command for the per-MB non-scaled coefficient data. Indirect data accessed at this address and beyond will return as 0 by the hardware. Setting this field to 0 will cause this range check to be ignored.If non-zero, this address must be greater than the MFD Indirect IT-COEFF Object Base Address state.Hardware ignores this field if indirect data is not present, i.e. the Indirect COEFF Data Length field of the MFD_IT_OBJECT command is set to 0.This field is only valid in MPEG2, AVC and VC1 decoder IT mode.
	11:0	**Reserved** Format: MBZ
15 **Project:** CHV, BSW	31:16	**Reserved** Project: All Format: MBZ Reserved for 64-bit address extension.
	15:0	**MFD Indirect IT-COEFF Object UpperBound [47:32]** Project: CHV, BSW **Description** This field is for the upper range of MFX Indirect IT-COEFF Object UpperBound. This field is for the upper range of MFX Indirect MV Object Base Address. This field is used for 48-bit addressing.
16	31:12	**MFD Indirect IT-DBLK Object - Base Address (Decoder Only)** Format: GraphicsAddress[31:12] Specifies the 4K-byte aligned memory base address for the read-only indirect data object pointed in the MFD_IT_OBJECT command for fetching (reading) the per-MB Deblocking filter control data.This field is only valid in AVC decoder IT mode.
	11:6	**Reserved** Format: MBZ
	5:0	**Reserved** Project: CHV, BSW Format: MBZ
17 **Project:** CHV, BSW	31:16	**Reserved** Project: CHV, BSW Format: MBZ Reserved for 64-bit address extension.

MFX_IND_OBJ_BASE_ADDR_STATE

	15:0	**MFD Indirect IT-DBLK Object Base Address [47:32]** Project: CHV, BSW **Description** / **Project** This field is for the upper range of MFX Indirect IT-DBLK Object Base Address. This field is used for 48-bit addressing. — CHV, BSW
18 **Project:** CHV, BSW	31:15	**Reserved** Project: CHV, BSW Format: MBZ
	14:13	**Reserved** Project: CHV, BSW Format: MBZ
	12:11	**Reserved** Project: CHV, BSW Format: MBZ
	10:9	**Reserved**
	8:7	**MFD Indirect IT-DBLK Object - Arbitration Priority Control** Project: CHV, BSW Format: U2 This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.
	6:5	**Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for MFD Indirect IT-DBLK ObjectBase Address** Project: CHV, BSW This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream.

Bits 8:7 values:

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

Bits 6:5 values:

Value	Name
00b	Use Cacheability Controls from page table
01b	Uncacheable (UC) - non-cacheable
10b	Writethrough (WT)
11b	Writeback (WB)

MFX_IND_OBJ_BASE_ADDR_STATE

	4:3	**Target Cache (TC) MFD Indirect IT-DBLK ObjectBase Address** Project: CHV, BSW This field allows the choice of LLC vs eLLC for caching **Value** / **Name** 00b: eLLC Only - not snooped in GT 01b: LLC Only 10b: LLC/eLLC Allowed 11b: L3, LLC, eLLC Allowed
	2	**Encrypted Data MFD Indirect IT-DBLK ObjectBase Address** Project: CHV, BSW Format: Enable This field controls whether data is decrypted while being read. This field is ignored for writes.
	1:0	**Age for QUADLRU (AGE) MFD Indirect IT-DBLK ObjectBase Address** Project: CHV, BSW Format: Enable This field allows the selection of AGE parameter for a given surface in LLC or eLLC. **Value** / **Name** 11b: Good chance of generating hits 10b: Next good chance of generating hits 01b: Decent chance of generating hits 00b: Poor chance of generating hits
19	31:12	**MFD Indirect IT-DBLK Object - Access Upper Bound (Decoder Only)** Format: GraphicsAddress[31:12] This field specifies the 4K-byte aligned (exclusive) maximum Graphics Memory address access by the indirect data object in the MFD_IT_OBJECT command for the per-MB Deblocking filter control data. Indirect data accessed at this address and beyond will return as 0 by the hardware. Setting this field to 0 will cause this range check to be ignored.If non-zero, this address must be greater than the MFD Indirect IT-DBLK Object Base Address state.Hardware ignores this field if indirect data is not present, i.e. the Indirect Deblocking Control Data Length field of the MFD_IT_OBJECT command is set to 0.This field is only valid in AVC decoder IT mode.
	11:0	**Reserved** Format: MBZ
20 **Project:** CHV, BSW	31:16	**Reserved** Project: All Format: MBZ Reserved for 64-bit address extension.

MFX_IND_OBJ_BASE_ADDR_STATE

DWord	Bit	Description
	15:0	**MFD Indirect IT-DBLK Object UpperBound [47:32]** Project: All **Description** / **Project** This field is for the upper range of MFX Indirect IT-DBLK Object UpperBound. This field is used for 48-bit addressing. — CHV, BSW
21	31:12	**MFC Indirect PAK-BSE Object - Base Address (Encoder Only)** Project: All Format: GraphicsAddress[31:12] Specifies the 4K-byte aligned memory base address for the write-only indirect data object pointed in the PAK_SLICE_STATE command for writing out the compressed bitstream.This field is only valid in AVC encoder mode.
	11:6	**Reserved** Project: All Format: MBZ
	5:0	**Reserved** Project: CHV, BSW Format: MBZ
22 **Project:** CHV, BSW	31:16	**Reserved** Project: CHV, BSW Format: MBZ Reserved for 64-bit address extension.
	15:0	**MFC Indirect PAK-BSE Object Base Address [47:32]** Project: CHV, BSW **Description** / **Project** This field is for the upper range of MFX Indirect PAK-BSE Object Base Address. This field is used for 48-bit addressing. — CHV, BSW
23 **Project:** CHV, BSW	31:15	**Reserved** Project: CHV, BSW Format: MBZ
	14:13	**Reserved** Project: CHV, BSW Format: MBZ
	12:11	**Reserved** Project: CHV, BSW Format: MBZ

MFX_IND_OBJ_BASE_ADDR_STATE

	10:9	**Reserved** Project: CHV, BSW Format: MBZ
	8:7	**MFC Indirect PAK-BSE Object Desitnation - Arbitration Priority Control** Project: CHV, BSW 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
	6:5	**Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for MFC Indirect PAK-BSE ObjectBase Address** Project: CHV, BSW This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream. Value / Name: 00b: Use Cacheability Controls from page table 01b: Uncacheable (UC) - non-cacheable 10b: Writethrough (WT) 11b: Writeback (WB)
	4:3	**Target Cache (TC) MFC Indirect PAK-BSE ObjectBase Address** Project: CHV, BSW This field controls the L3$, LLC and eLLC (eDRAM) cacheability for a given surface. Setting of "00" points to PTE settings which defaults to eDRAM (when present). If no eDRAM, the access will be allocated to LLC. Setting of "01", allocates into LLC and victimizes the line to eDRAM. Setting of "10" allows the line to be allocated in either LLC or eDRAM. Setting of "11" is the only option for a memory access to be allocated in L3$ as well as LLC/eLLC 00b: eLLC Only *("00" setting points TC selection to PTE which defaults to eLLC)* 01b: LLC Only (*Works at the allocation time, later victimization from LLC downgrades the line to eLLC if present*). 10b: LLC/eLLC Allowed. 11b: L3, LLC, eLLC Allowed. **Errata CHV:A-E (FIXED BY:G0 Stepping):** For all system that does NOT use SVM (i.e. coherent L3$ surfaces), back snoops from LLC has to be disabled (**Dis_GtCvUpdtOnRd = “1”**). Than target Cache settings can be programmed as POR requirements of L3/LLC/eDRAM caching. For all systems that does use SVM (i.e. coherent L3$ surfaces), the recomended setting would

MFX_IND_OBJ_BASE_ADDR_STATE

		be "00" in target cache settings. In case of L3 surfaces, the performance has to be tuned between "00" and "11" setting based on the benefits of L3 caching outweighting the degradation of backsnoops. Post G0-stepping, the above w/a for coherent L3$ surfaces is not needed. **Value** / **Name** 00b — eLLC Only - not snooped in GT 01b — LLC Only 10b — LLC/eLLC Allowed 11b — L3, LLC, eLLC Allowed
	2	**Encrypted Data MFC Indirect PAK-BSE ObjectBase Address** Project: CHV, BSW Format: Enable This field controls whether data is decrypted while being read. This field is ignored for writes.
	1:0	**Age for QUADLRU (AGE) MFC Indirect PAK-BSE ObjectBase Address** Project: CHV, BSW Format: Enable This field allows the selection of AGE parameter for a given surface in LLC or eLLC. If a particular allocation is done at youngest age ("0,1,2") it tends to stay longer in the cache. This option is given to GFX software to be able to decide which surfaces are more likely to generate HITs, hence need to be replaced least often in caches. **Value** / **Name** 11b — Good chance of generating hits 10b — Next good chance of generating hits 01b — Decent chance of generating hits 00b — Poor chance of generating hits
24 **Project:** CHV, BSW	31:12	**MFC Indirect PAK-BSE Object - Access Upper Bound (Eecoder Only)** Project: CHV, BSW Format: GraphicsAddress[31:12] This field specifies the 4K-byte aligned (exclusive) maximum Graphics Memory address access by the indirect data object in the PAK_SLICE_STATE command for the per-slice output bitstream. Indirect data accessed at this address and beyond will be blocked by the hardware and ignored. Setting this field to 0 will cause this range check to be ignoredIf non-zero, this address must be greater than the MFC Indirect PAK-BSE Object Base Address state.This field is only valid in AVC encoder mode. **Programming Notes** / **Project** For VP8 Encoder, this field should be programmed the same at both DW4, DW5 **MFX Indirect Bitstream Object - Access Upper Bound** as well as DW24, DW25. — CHV, BSW
	11:0	**Reserved**

MFX_IND_OBJ_BASE_ADDR_STATE		
25 **Project:** CHV, BSW	31:16	**Reserved** <table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table>Reserved for 64-bit address extension.
	15:0	**MFC Indirect PAK-BSE Object UpperBound [47:32]** <table><tr><td>Project:</td><td>CHV, BSW</td></tr></table><table><tr><th>Description</th><th>Project</th></tr><tr><td>This field is for the upper range of MFC Indirect PAK-BSE Object UpperBound</td><td></td></tr><tr><td>This field is used for 48-bit addressing.</td><td>CHV, BSW</td></tr></table><table><tr><th>Programming Notes</th><th>Project</th></tr><tr><td>For VP8 Encoder, this field should be programmed the same at both DW4, DW5 MFX Indirect Bitstream Object - Access Upper Bound as well as DW24, DW25.</td><td>CHV, BSW</td></tr></table>

MFX_JPEG_HUFF_TABLE_STATE

MFX_JPEG_HUFF_TABLE_STATE

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_MULTI_DW Format: OpCode
	26:24	**Media Command Opcode** Default Value: 7h JPEG_COMMON Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 2h Format: OpCode
	15:12	**Reserved**
	11:0	**DWord Length** Default Value: 033Dh Excludes DWord (0,1) Project: All Format: =n Total Length - 2
1	31:1	**Reserved**
	0	**HuffTableID (1-bit)** Identifies the huffman table. Value: 0 / Name: Y / Description: Huffman table for Y

<table>
<tr><th colspan="3">MFX_JPEG_HUFF_TABLE_STATE</th></tr>
<tr><td>2..4</td><td>31:0</td><td>DC_BITS (12 8-bit array)
The number of DC Huffman codes of length i, where i is 1~12</td></tr>
<tr><td>5..7</td><td>31:0</td><td>DC_HUFFVAL (12 8-bit array)
The value associated with each DC Huffman code of length i.</td></tr>
<tr><td>8..11</td><td>31:0</td><td>AC_BITS (16 8-bit array)
the list of Li, number of Huffman codes of length i, where i is 1~16</td></tr>
<tr><td>12..51</td><td>31:0</td><td>AC_HUFFVAL (160 8-bit array)
the list of Vi,j, the value associated with each Huffman code of length i</td></tr>
<tr><td rowspan="2">52</td><td>31:16</td><td>Reserved<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>15:0</td><td>AC_HUFFVAL(2-8 bit array)
In AC table, BITS can have up to 16-bit codeword. Li can be 0 ~ 162. HUFFVAL will have a list of likely random distributed values</td></tr>
</table>

MFX_JPEG_PIC_STATE

MFX_JPEG_PIC_STATE	
Project:	CHV, BSW
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 MFX_MULTI_DW Format: OpCode
	26:24	**Media Command Opcode** Default Value: 7h JPEG Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Common Format: OpCode
	20:16	**SubOpcode B** Default Value: 0h MEDIA_ Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Project: All Format: =n Total Length - 2 Value: 0001h — Name: **[Default]** — Description: Excludes DWord (0,1)
1	31	**Reserved**
	30:26	**Pixels In Horizontal Last MCU** Project: CHV, BSW Exists If: //Encoder Only The number of pixels in the last MCU in a row MCUs. This information is used for completion of partial MCU.

MFX_JPEG_PIC_STATE

	31:21	**Reserved** Exists If: //Decoder Only Format: MBZ
	25:21	**Pixels In Vertical Last MCU** Project: CHV, BSW Exists If: //Encoder Only The number of pixels in the last MCU in a column MCUs. This information is used for completion of partial MCU.
	20	**Vertical Up-Sampling Enable** Project: CHV, BSW Exists If: //Decoder Only Only applied to chroma blocks. This flag is used for 2:1 vertical up-sampling for chroma 420 and outputting chroma422 YUY2 or UYVY format. To enable this flag, the input should be interleaved Scan, **InputFormatYUV** should be set to YUV420, and **OutputFormatYUV** should be set to YUY2 or UYVY. Value / Name / Description: 0b — no up-sampling 1b — 2:1 vertical up-sampling
	19	**Reserved** Project: CHV, BSW Exists If: //Decoder Only
	18	**Horizontal Down-Sampling Enable** Project: CHV, BSW Exists If: //Decoder Only Only applied to chroma blocks. This flag is used for 2:1 horizontal down-sampling for chroma 422 and outputting chroma420 NV21 format. To enable this flag, the input should be interleaved Scan, **InputFormatYUV** should be set to YUV422V_2Y or YUV422V_4Y, and **OutputFormatYUV** should be set to NV12. Value / Name / Description: 0b — no down-sampling 1b — 2:1 horizonatl down-sampling
	17	**Vertical Down-Sampling Enable** Project: CHV, BSW Exists If: //Decoder Only Only applied to chroma blocks. This flag is used for 2:1 vertical down-sampling for chroma 422 and outputting chroma420 NV21 format. To enable this flag, the input should be interleaved Scan, **InputFormatYUV** should be set to YUV422H_2Y or YUV422H_4Y, and **OutputFormatYUV** should be set to NV12.

MFX_JPEG_PIC_STATE

Value	Name	Description
0b		no down-sampling
1b		2:1 vertical down-sampling

16 — Average Down Sampling

Project:	CHV, BSW
Exists If:	//Decoder Only

This flag is used to select a down-sampling method when **VertDownSamplingEnb** or **HoriDownSamplingEnb** is set to 1.

Value	Name	Description
0b		Drop every other line (or column) pixels
1b		Average neighboring two pixels

20:12 — Reserved

Project:	CHV, BSW
Exists If:	//Encoder Only
Format:	MBZ

15:12 — Reserved

Exists If:	//Decoder Only
Format:	MBZ

11:8 — Output Format YUV

Project:	CHV, BSW
Exists If:	//Decoder Only

This field specifies the surface format to write the decoded JPEG image.Note that any non-interleaved JPEG input should be set to "0000". For the interleaved input Scan data, it can be set either "0000" or the corresponding format.

Value	Name	Description
0000b		3 separate plane for Y, U, and V respectively
0001b		NV12 for chroma 4:2:0
0010b		UYVY for chroma 4:2:2
0011b		YUY2 for chroma 4:2:2

Programming Notes

The **MFX_SURFACE_STATE** command should be set accordingly for each **OutputFormatYUV**.
For NV12, **Surface Format** = 4 (PLANAR_420_8)
For YUY2, **Surface Format** = 0 (YCRCB_NORMAL)
For UYVY, **Surface Format** = 3 (YCRCB_SWAPY)
NV12 (0001b) can be set only when Y, U, V are interleaved in a single Scan data with the following cases

- **InputFormatYUV** is YUV420 and **VertDownSamplingEnb** is disalbed

MFX_JPEG_PIC_STATE

- **InputFormatYUV**is YUV422H_2Y or YUV422H_4Y, and **VertDownSamplingEnb** is enabled

UYVY (0010b) and YUY2 (0011b) can be set only when Y, U, V are interleaved in a single Scan data with the following cases

- **InputFormatYUV** is YUV420 and **VertUpSamplingEnb** is enabled
- **InputFormatYUV** is YUV422H_2Y or YUV422H_4Y and **VertUpSamplingEnb** is disabled

11:8 **Input Surface Format YUV**

Project:	CHV, BSW
Exists If:	//Encoder Only

This field specifies the surface format to read a YUV image data

Value	Name	Description
0000b		Reserved
0001b	NV12	NV12 for chroma 4:2:0
0010b	UYVY	UYVY for chroma 4:2:2
0011b	YUY2	YUY2 for chroma 4:2:2
0100b	Y8	Y8 for chroma400 Y-only image
0101b	RGB	RGB or YUV for chroma 4:4:4

Programming Notes
This field should be set accordingly for **SurfaceFormat** in MFX_SURFACE_STATE command.
R8G8B8A8_UNORM in this field is used for encoding RGB and YUV chroma 4:4:4. For RGB input, any order of image components R, G, B (e.g., RGB, GBR, BGR, YUV) will be acceptable as far as the order of Quantization tables and Huffman tables match the order of image components.

7:6 **Reserved**

Exists If:	//Decoder Only
Format:	MBZ

7:6 **Reserved**

Project:	CHV, BSW
Exists If:	//Encoder Only
Format:	MBZ

5:4 **Rotation**

Exists If:	//Decoder Only

Value	Name	Description
00b		no rotation
01b		rotate clockwise 90 degree
10b		rotate counter-clockwise 90 degree (same as rotating 270 degree clockwise)

MFX_JPEG_PIC_STATE

11b		rotate 180 degree (NOT the same as flipped on the x-axis)

Programming Notes	Project
Rotation can be set to 01b, 10b, or 11b when OutputFormatYUV is set to 0000b. For other OutputFormatYUV, Rotation is not allowed.	CHV, BSW

5:3 **Reserved**

Project:	CHV, BSW
Exists If:	//Encoder Only
Format:	MBZ

3 **Reserved**

Exists If:	//Decoder Only
Format:	MBZ

2:0 **Input Format YUV**

Exists If:	//Decoder Only
Format:	U3

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

2:0 **Output MCU Structure**

Project:	CHV, BSW
Exists If:	//Encoder Only

Output MCU Structure(**OutputMcuStructure**) should be set accordingly for each Input Surface Format YUV(**InputSurfaceFormatYUV**):

- If **InputSurfaceFormatYUV** is set to NV12, **OutputMCUStructure** is set to YUV420.
- If **InputSurfaceFormatYUV** is set to UYVY or YUY2, **OutputMCUStructure** is set to YUV422H_2Y.
- If **InputSurfaceFormatYUV** is set to Y8, **OutputMCuStructure** is set to YUV400.
- If **InputSurfaceFormatYUV** is set to RGB (or GBR, BGR, YUV), **OutputMCuStructure** is set to RGB.

MFX_JPEG_PIC_STATE

		• If **InputSurfaceFormatYUV** is set to RGB, the order of encoded blocks in MCU will be same as the order of input image components. If the order of input image components is RGB (or GBR, BGR, YUV), then the order of blocks will be RGB (or GBR, BGR, YUV respectively).

Value	Name	Description
0	YUV400	Grayscale Image
1	YUV420	Both horizontally and vertically chroma 2:1 subsampled
2	YUV422H_2Y	Horizontally chroma 2:1 subsampled - horizontal 2 Y-blocks, 1 U and 1 V block
3	RGB	RGB or YUV444: No subsample
4		
5		
6		
7		

DWord	Bit	Description
2	31:30	**Reserved**

Exists If:	//Decoder Only
Format:	MBZ

DWord	Bit	Description
	31:29	**Reserved**

Project:	CHV, BSW
Exists If:	//Encoder Only
Format:	MBZ

DWord	Bit	Description
	29	**Reserved**

Exists If:	//Decoder Only
Format:	MBZ

DWord	Bit	Description
	28:16	**Frame Height In Blocks Minus 1**

Exists If:	//Decoder Only
Format:	U13-1

Description
(The number of blocks in height) - 1. This value is calculated using the number of lines Y and vertical sampling factor of the first component V_1 in Frame header. See the note following this table. For interleaved components, (((Y + (V_1*8 -1)) / (V_1*8)) * V_1) - 1, where "/" is integer division. For non-interleaved components, ((Y + 7) / 8) - 1.

Workaround: For interleaved components, when **Input Format YUV** is set to **YUV422H_2Y**, **OutputFormatYUV** is set to **NV12**,
If ((((((Y + (V_1*8 -1)) / (V_1*8)) * V_1) - 1)% 2) == 0),
then **Frame Height In Blocks Minus 1** = ((Y + (V1*8 -1)) / (V1*8)) * V1
else

MFX_JPEG_PIC_STATE

		then **Frame Height In Blocks Minus 1** = (((Y + (V1*8 -1)) / (V1*8)) * V1) - 1
	28:16	**Frame Height In Blks Minus 1** Project: CHV, BSW Exists If: //Encoder Only Format: U13-1 (The number of blocks in height) - 1. This value is calculated using the number of lines Y and vertical sampling factor of the first component V1 in Frame header. See the note following this table. For interleaved components: (((Y + (V1*8 -1)) / (V1*8)) * V1) - 1 For non-interleaved components: ((Y + 7) / 8) - 1
	15:13	**Reserved** Exists If: //Decoder Only Format: MBZ
	15:13	**RoundingQuant** Project: CHV, BSW Exists If: //Encoder Only Rounding value applied to quantization output (see value table below)
	12:0	**Frame Width In Blocks Minus 1** Exists If: //Decoder Only Format: U13-1 (The number of blocks in width) - 1. This value is calculated using the number of samples per line X and horizontal sampling factor of the first component H_1 in Frame header. See the note following this table. For interleaved components, (((X + (H_1 *8 -1)) / (H_1 *8)) * H_1) - 1. For non-interleaved components, ((X + 7) / 8) - 1.
	12:0	**Frame Width In Blks Minus 1** Project: CHV, BSW Exists If: //Encoder Only Format: U13-1

RoundingQuant values:

Value	Name	Description
000b	**[Default]**	1/2
001b		(1/2 - 1/128)
010b		(1/2 + 1/128)
011b		(1/2 - 1/64)
100b		(1/2 + 1/64)
101b		(1/2 - 1/32)
110b		(1/2 - 1/16)
111b		(1/2 - 1/8)

MFX_JPEG_PIC_STATE		
		(The number of blocks in width) - 1. This value is calculated using the number of samples per line X and horizontal sampling factor of the first component H1 in Frame header. See the note following this table.
		For interleaved components: (((X + (H1 *8 -1)) / (H1 *8)) * H1) - 1 For non-interleaved components: ((X + 7) / 8) - 1

MFX_MPEG2_PIC_STATE

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

MFX_MPEG2_PIC_STATE

	23:20	**f_code[0][1]** Used for forward motion vector prediction. See ISO/IEC 13818-2 7.6.3.1 for details
	19:16	**f_code[0][0]** Used for forward motion vector prediction. See ISO/IEC 13818-2 7.6.3.1 for details
	15:14	**Intra DC Precision** Project: All Format: U2 See ISO/IEC 13818-2 6.3.10 for details.
	13:12	**Picture Structure** This field specifies whether the picture is encoded in the form of a frame picture or one field (top or bottom) picture. See ISO/IEC 13818-2 6.3.10 for details.Format = MPEG_PICTURE_STRUCTURE00 = Reserved01 = MPEG_TOP_FIELD10 = MPEG_BOTTOM_FIELD11 = MPEG_FRAME
	11	**TFF (Top Field First)** When two fields are stored in a picture, this bit indicates if the top field is the first field.For a frame P picture, the value 1 indicates that the top field of the reconstructed frame is the first field output by the decoding process, the same as defined in ISO/IEC 13818-2 6.3.10. Particularly, it is used by the hardware to calculate derivative motion vectors from the dual-prime motion vectors.For a field P picture, hardware uses this bit together with the Picture Structure to determine if the current picture is the Second Field. In this case, the definition of this bit differs from ISO/IEC 13818-2 6.3.10 - software must derive the value for this bit according to the following relation:Picture Structure = top fieldPicture Structure = bottom fieldSecond Field = 0TFF = 1TFF = 0Second Field = 1TFF = 0TFF = 1
	10	**Frame Prediction Frame DCT** This field provides constraints on the DCT type and prediction type. It affects the syntax of the bitstream.
	9	**Concealment Motion Vector Flag** This field indicates if the concealment motion vectors are coded in intra macroblocks. It affects the syntax of the bitstream.
	8	**Quantizer Scale Type** Format: MPEG_Q_SCALE_TYPE This field specifies the quantizer scaling type. Value | Name | Description 0h | | MPEG_QSCALE_LINEAR 1h | | D MPEG_QSCALE_NONLINEAR esc
	7	**Intra VLC Format** This field is used by VLD
	6	**Scan Order** Format: MPEG_INVERSESCAN_TYPE

MFX_MPEG2_PIC_STATE

<table>
<tr><td></td><td></td><td>This field specifies the Inverse Scan method for the DCT-domain coefficients in the blocks of the current picture.
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td></td><td>MPEG_ZIGZAG_SCAN</td></tr><tr><td>1h</td><td></td><td>MPEG_ALTERNATE_VERTICAL_SCAN</td></tr></table></td></tr>
<tr><td></td><td>5:0</td><td>Reserved</td></tr>
<tr><td>2</td><td>31</td><td>I Slice Concealment Mode
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Exists If:</td><td>//Decoder</td></tr></table>This field controls how MPEG decoder handles MB concealment in I Slice
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Intra Concealment</td><td>Using Coefficient values to handle MB concealment</td></tr><tr><td>1h</td><td>Inter Concealment</td><td>Using Motion Vectors to handle MB concealment</td></tr></table><table><tr><th>Programming Notes</th></tr><tr><td>If this field is set to "1", driver must provide a valid forward reference picture (both top and bottom Field must be valid)</td></tr></table></td></tr>
<tr><td></td><td>30</td><td>Reserved
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>29:28</td><td>P/B Slice Concealment Mode
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Exists If:</td><td>//Decoder</td></tr></table>This field controls how MPEG decoder handles MB concealment in P/B Slice.
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>00b</td><td>INTER</td><td>If left MB is NOT Intra MB type (including skipMB), use left MB inter prediction mode [frame/field or forward/backward/bi] and MV final values as concealment. Otherwise (left MB is Intra MB), use forward reference (same polarity for field pic) with MV final values set to 0.</td></tr><tr><td>01b</td><td>LEFT</td><td>If left MB is NOT Intra MB type (including skipMB), use left MB inter prediction mode [frame/field or forward/backward/bi] and MV final values as concealment. Otherwise (left MB is Intra MB), use left MB dct_dc_pred[cc] values for concealment (Macroblock is concealed as INTRA MB and dct_dc_pred[cc] are DC predictor for Luma, Cr, Cb data)</td></tr><tr><td>10b</td><td>ZERO</td><td>Always use forward reference (same polarity for field pic) with MV final values set to 0 (Macroblock is concealed as INTER coded)</td></tr><tr><td>11b</td><td>INTRA</td><td>Use left MB dct_dc_pred[cc] values for concealment (Macroblock is concealed as INTRA MB and dct_dc_pred[cc] are DC predictor for Luma, Cr, Cb data</td></tr></table></td></tr>
</table>

MFX_MPEG2_PIC_STATE

	27	**Reserved** Project: CHV, BSW Format: MBZ
	26:25	**P/B Slice Predicted BiDir Motion Type Override - Bi-direction MV Type Override** Project: CHV, BSW Exists If: //Decoder This field is only applicable if the Concealment Motion Type is predicted to be Bi-directional. (It is only possible if "P/B Slice Concealment Mode" is set to "00" or "01" and left MB is a bi-directional MB). **Value / Name / Description** 0h / BID / Keep Bi-direction Prediction 1h / RESERVED / 2h / FWD / Only use Forward Prediction (Backward MV is forced to invalid 3h / BWD / Only use Backward Prediction (Forward MV is forced to invalid)
	24	**P/B Slice Predicted Motion Vector Override Final MV value Override** Project: CHV, BSW Exists If: //Decoder This field is only applicable if the Concealment Motion Vectors are non-zero. It is only possible if "P/B Slice Concealment Mode" is set to "00" or "01" and left MB has non-zero motion vectors). **Value / Name / Description** 0h / Predicted / Motion Vectors use predicted values 1h / ZERO / Motion Vectors force to 0
	23:15	**Reserved** Format: MBZ
	14	**LoadSlicePointerFlag - LoadBitStreamPointerPerSlice** Exists If: //Encoder To support multiple slice picture and additional header/data insertion before and after an encoded slice.When this field is set to 0, bitstream pointer is only loaded once for the first slice of a frame. For subsequent slices in the frame, bitstream data are stitched together to form a single output data stream.When this field is set to 1, bitstream pointer is loaded for each slice of a frame. Basically bitstream data for different slices of a frame will be written to different memory locations. **Value / Name / Description** 0h / / Load BitStream Pointer only once for the first slice of a frame 1h / / Load/reload BitStream Pointer only once for the each slice, reload the start location of the bitstream buffer from the Indirect PAK-BSE Object Data Start Address field
	13	**Reserved**

MFX_MPEG2_PIC_STATE

DWord	Bit	Description
	12	**Reserved** Format: MBZ
	11	**Reserved** Format: MBZ
	10:9	**Picture Coding Type** Format: MPEG_PICTURE_CODING_TYPE This field identifies whether the picture is an intra-coded picture (I), predictive-coded picture (P) or bi-directionally predictive-coded picture (B). See ISO/IEC 13818-2 6.3.9 for details. Value / Name: 00b: Reserved 01b: MPEG_I_PICTURE 10b: 10 = MPEG_P_PICTURE 11b: MPEG_B_PICTURE
	8:2	**Reserved** Format: MBZ
	1	**MismatchControlDisabled** These 2 bits flag disables mismatch control of the inverse transformation for some specific cases during reference reconstruction. Value / Name / Description: 00b: Mismatch control applies to all MBs 01b: Disable mismatch control to all intra MBs whose all AC-coefficients are zero. 10b: Disable mismatch control to all MBs whose all AC-coefficients are zero. 11b: Disable mismatch control to all MBs.
	0	**Disable Mismatch** To disable MPEG2 IDCT fixed point arithmetic correction
3	31	**Slice Concealment Disable Bit** Project: CHV, BSW Exists If: //Decode If VINunit detects the next slice starting position is either out-of-bound or smaller than or equal to the current slice starting position, VIN will set the current slice to be 1 MB and force VMDunit to do slice concealment on the next slice. This bit will disable this feature and the MB data from the next slice will be decoded from bitstream. Value / Name / Description: 0h: Enable **[Default]** — VIN will force next slice to be concealment if detects slice boundary error 1h: Disable — VIN will not force next slice to be in concealment

MFX_MPEG2_PIC_STATE

<table>
<tr><td></td><td></td><td>Programming Notes
Driver has an option to detect the scenario given in description (above) and remove the second (out-of-order) slice. In this case, hardware will decode the first slice in completion and do concealment till the third slice. It should yield a picture with better quality this way.</td></tr>
<tr><td></td><td>30:29</td><td>Reserved
Format: MBZ</td></tr>
<tr><td></td><td>28:24</td><td>Reserved</td></tr>
<tr><td></td><td>23:16</td><td>FrameHeightInMBsMinus1[7:0] (Picture Height in Macroblocks)
Format: U8</td></tr>
<tr><td></td><td>15:8</td><td>Reserved
Format: MBZ for future supporting width > 4K</td></tr>
<tr><td></td><td>7:0</td><td>FrameWidthInMBsMinus1[7:0] (Picture Width in Macroblocks)
Project: All
Format: U8</td></tr>
<tr><td>4</td><td>31:16</td><td>MinFrameWSize
Project: All
Format: U16
- Minimum Frame Size [15:0] (16-bit) (Encoder Only)Mininum Frame Size is specified to compensate for intel Rate ControlCurrently zero fill (no need to perform emulation byte insertion) is done only to the end of the CABAC_ZERO_WORD insertion (if any) at the last slice of a picture. Intel encoder parameter, not part of DXVA. The caller should always make sure that the value, represented by Mininum Frame Size, is always less than maximum frame size FrameBitRateMax (DWORD 10 bits 29:16). This field is reserved in Decode mode.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>[0,0003FFFFh]</td><td></td><td>The programmable range when MinFrameWSizeUnits is 00.</td></tr>
<tr><td>[0,000FFFFFh]</td><td></td><td>The Programmable range when MinFrameWSizeUnits is 01.</td></tr>
<tr><td>[0,03FFFFFFh]</td><td></td><td>The Programmable range when MinFrameWSizeUnits is 10.</td></tr>
<tr><td>[0, FFFFFFFFh]</td><td></td><td>The Programmable range when MinFrameWSizeUnits is 11.</td></tr>
<tr><td>0h</td><td>[Default]</td><td></td></tr>
</table></td></tr>
<tr><td></td><td>15</td><td>Reserved
Project: All
Format: MBZ</td></tr>
<tr><td></td><td>14:12</td><td>RoundInterAC,
rounding precision for non-Intra AC000: +1/16001: +2/16010: +3/16011: +4/16100: +5/16101: +6/16110: +7/16111: +8/16</td></tr>
<tr><td></td><td>11</td><td>Reserved</td></tr>
</table>

MFX_MPEG2_PIC_STATE

	10:8	**RoundIntraAC** Project: All Format: U3 rounding precision for Intra AC000: +1/16001: +2/16010: +3/16011: +4/16100: +5/16101: +6/16110: +7/16111: +8/16
	7	**Reserved** Format: MBZ
	6:4	**RoundInterDC** rounding Precision for non-Intra-DC000: +1/16001: +2/16010: +3/16011: +4/16100: +5/16101: +6/16110: +7/16111: +8/16
	3	**Reserved** Format: MBZ
	2:1	**RoundIntraDC** rounding Precision for Intra-DC00: +1/801: +2/810: +3/811: +4/8
	0	**Reserved**
5	31:17	**Reserved** (for future Mask bits)
	16	**FrameSizeControlMask** Frame size conformance maskThis field is used when MacroblockStatEnable is set to 1. Value | Name | Description 0h | | Do not change Slice Quantization Parameter values in MFC_MPEG2_SLICEGROUP_STATE with suggested slice QP value for frame level Rate control 1h | | Replace Slice Quantization Parameter values in MFC_MPEG2_SLICEGROUP_STATE with suggested slice QP value for frame level Rate control values in MFC_IMAGE_STATUS control register.
	15:13	**Reserved**
	12	**InterMBForceCBPZeroControlMask** Format: U1 Inter MB Force CBP ZERO mask. Value | Name | Description | Project [0, FFFFFFFFh] | | | 0h | | No effect | All 1h | | Zero out all A/C coefficients for the inter MB violating Inter Confirmance | All
	11:10	**MinFrameWSizeUnits** This field is the Minimum Frame Size Units

MFX_MPEG2_PIC_STATE

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 **MBRateControlMask**

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

Value	Name	Description
0h		Do not change QP values of inter macroblock with suggested QP values in Macroblock Status Buffer
1h		Apply RC QP delta for all macroblock

8 **Reserved**

7 **Reserved**

6:4 **Reserved**

3 **FrameBitRateMinReportMask**

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

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

2 **FrameBitRateMaxReportMask**

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

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

1 **InterMBMaxSizeReportMask**

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

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

<table>
<tr><th colspan="3">MFX_MPEG2_PIC_STATE</th></tr>
<tr><td></td><td>0</td><td>IntraMBMaxSizeReportMask
This is a mask bit controlling if the condition of any intra MB in the frame exceeds IntraMBMaxSize.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td></td><td>Do not update bit0 of MFC_IMAGE_STATUS control register.</td></tr>
<tr><td>1h</td><td></td><td>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.</td></tr>
</table></td></tr>
<tr><td rowspan="4">6

[ExistsIf]Encode Only</td><td>31:28</td><td>Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>27:16</td><td>InterMBMaxSize
<table><tr><td>Default Value:</td><td>FFFh</td></tr></table>
This field, Inter MB Conformance Max size limit,indicates the allowed max bit count size for Inter MB</td></tr>
<tr><td>15:12</td><td>Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>11:0</td><td>IntraMBMaxSize
<table><tr><td>Default Value:</td><td>FFFh</td></tr></table>
This field, Intra MB Conformance Max size limit,indicates the allowed max bit count size for Intra MB</td></tr>
<tr><td rowspan="2">7</td><td>31:1</td><td>Reserved</td></tr>
<tr><td>0</td><td>VSL top MB Trans8x8flag
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Exists If:</td><td>//Encode Only</td></tr></table>
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>Disable</td><td>VSL will only fetch the current MB data.</td></tr>
<tr><td>1</td><td>Enable</td><td>When this bit is set VSL will make extra fetch to memory to fetch the MB data for top MB.</td></tr>
</table></td></tr>
<tr><td>8

[ExistsIf]Encode Only</td><td>31:24</td><td>SliceDeltaQPMax[3]
<table><tr><td>Format:</td><td>S7</td></tr></table>
<table><tr><td>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).</td></tr><tr><td>Range: [-30,30]</td></tr></table></td></tr>
</table>

MFX_MPEG2_PIC_STATE

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

MFX_MPEG2_PIC_STATE

	23:16	**SliceDeltaQPMin[2]** Format: S7 Range: [-30,30] This field is the Slice level delta QP for bit-count below FrameBitRateMin - below 1/ 8 and above 1/ 4This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between one-eighth and quarter the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of [(FrameBitRateMin- FrameBitRateMinDelta»2), (FrameBitRateMin- FrameBitRateMinDelta»3)).
	15:8	**SliceDeltaQPMin[1]** Format: S7 Range: [-30,30] This field is the Slice level delta QP for bit-count below FrameBitRateMin- below 1/4 and above 1/ 2This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between quarter and half the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of [(FrameBitRateMin- FrameBitRateMinDelta»1), (FrameBitRateMin- FrameBitRateMinDelta»2)).
	7:0	**SliceDeltaQPMin[0]** Format: S7 Range: [-30,30] This field is the Slice Level Delta QP for bit-count below FrameBitRateMin - below 1/ 2This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is below FrameBitRateMin by more than half the distance of FrameBitRateMinDelta , i.e., in the range of [0, (FrameBitRateMin- FrameBitRateMinDelta»1).
10 [ExistsIf]Encode Only	31	**FrameBitrateMaxUnit** This field is the Frame Bitrate Maximum Limit Units.

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

MFX_MPEG2_PIC_STATE

DWord	Bit	Description
	30	**FrameBitrateMaxUnitMode**

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

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

DWord	Bit	Description
	29:16	**FrameBitRateMax**

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

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

DWord	Bit	Description
	15	**FrameBitrateMinUnit**

This field is the Frame Bitrate Minimum Limit Units.

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

DWord	Bit	Description
	14	**FrameBitrateMinUnitMode**

This field is the Frame Bitrate Minimum Limit Units.ValueNameDescriptionProject

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

DWord	Bit	Description
	13:0	**FrameBitRateMin**

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

DWord	Bit	Description
11	31	**Reserved**

MFX_MPEG2_PIC_STATE		
[ExistsIf]Encode Only	30:16	**FrameBitRateMaxDelta** Default Value: 0h Project: All Access: None Format: U15 This field is used to select the slice delta QP when FrameBitRateMax Is exceeded. It shares the same FrameBitrateMaxUnit. The programmable range is either 0- 512KB or 4MBB in FrameBitrateMaxUnit of 128 Bytes or 16KB respectively. This field is used to select the slice delta QP when FrameBitRateMax Is exceeded. It shares the same FrameBitrateMaxUnit. When FrameBitrateMaxUnitMode is 0(compatibility mode) bits 16:27 should be used, bits 28, 29 and 30 should be 0.
	15	**Reserved** Project: All Format: MBZ
	14:0	**FrameBitRateMinDelta** This field is used to select the slice delta QP when FrameBitRateMin Is exceeded. It shares the same FrameBitrateMinUnit. When FrameBitrateMinUnitMode is 0(compatibility mode) bits 0:11 should be used, bits 12, 13 and 14 should be 0.Note: HW requires the following condition FrameBitRateMinDelta <= 2*FrameBitRateMinMust be true, otherwise it may cause unpredicted behavior. Value / Name / Description: 0-1024KB / / The programmable range 0-1024KB When FrameBitrateMinUnit is in 32Bytes. 0-16380KB / / Programmable range is 0-16380KB when FrameBitrateMinUnit is in 4Kbytes.
12	31:21	**Reserved** Format: MBZ
	20	**VMD Error Logic** Project: CHV, BSW Value / Name / Description: 0 / Disable **[Default]** / 1 / Enable / Error Handling
	19	**Reserved** Format: MBZ
	18	**VAD Error Logic** Project: CHV, BSW

MFX_MPEG2_PIC_STATE		
		<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td>Enable **[Default]**</td><td>Error reporting ON in case of premature Slice done</td></tr><tr><td>1</td><td>Disable</td><td>CABAC Engine will auto decode the bitstream in case of premature slice done.</td></tr></table>
	17	**Reserved** Project: CHV, BSW
	16	**Reserved** Project: CHV, BSW
	15:0	**Reserved** Format: MBZ

MFX_PAK_INSERT_OBJECT

MFX_PAK_INSERT_OBJECT	
Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

Description
The MFX_PAK_INSERT_OBJECT command is the first primitive command for the AVC, MPEG2 and SVC Encoding Pipeline. The MFX_PAK_INSERT_OBJECT command is the first primitive command for the AVC, MPEG2, JPEG, SVC and VP8 Encoding Pipeline.
This command is issued to setup the control and parameters of inserting a chunk of compressed/encoded bits into the current bitstream output buffer starting at the specified bit locationto perform the actual insertion by transferring the command inline data to the output buffer max, 32 bits at a time. It is a variable length command as the data to be inserted are presented as inline data of this command. It is a multiple of 32-bit (1 DW), as the data bus to the bitstream buffer is 32-bit wide. Multiple insertion commands can be issued back to back in a series. It is host software's responsibility to make sure their corresponding data will properly stitch together to form a valid H.264 bitstream. Internally, MFX hardware will keep track of the very last two bytes' (the very last byte can be a partial byte) values of the previous insertion. It is required that the next Insertion Object Command or the next PAK Object Command to perform the start code emulation sequence check and prevention 0x03 byte insertion with this end condition of the previous insertion. Hardware will keep track of an output bitstream buffer current byte position and the associated next bit insertion position index. Data to be inserted can be a valid H.264 NAL units or a partial NAL unit. Certain NAL unit has a minimum byte size requirement. As such the hardware will optionally (enabled by STATE Command) determines the number of CABAC_ZERO_WORD to be inserted to the end of the current NAL, based on the minimum byte size of a NAL and the actual bin count of the encoded Slice. Since prior to the CABAC_ZERO_WORD insertion, the RBSP or EBSP is already byte-aligned, so each CABAC_ZERO_WORD insertion is actually a 3-byte sequence 0x00 00 03. The inline data may have already been processed for start code emulation byte insertion, except the possibility of the last 2 bytes plus the very last partial byte (if any). Hence, when hardware performing the concatenation of multiple consecutive insertion commands, or concatenation of an insertion command and a PAK object command, it must check and perform the necessary start code emulation byte insert at the junction.The inline data is required to be byte aligned on the left (first transmitted bit order) and may or may not be byte aligned on the right (last transmitted bits). The command will specify the bit offset of the last valid DW.Each insertion state command defines a chunk of bits (compressed data) to be inserted at a specific location of the output compressed bitstream in the output buffer.Depend on CABAC or CAVLC encoding mode (from Slice State), PAK Object Command is always ended in byte aligned output bitstream except for CABAC header insertion which is bit aligned. In the aligned cases, PAK will perform 0 filling in CAVLC mode, and 1 filling in CABAC mode. Insertion data can include:any encoded syntax elements bit data before the encoded Slice Data (PAK Object Command) of the current SliceSPS NALPPS NALSEI NALOther Non-Slice NALLeading_Zero_8_bits (as many bytes as there is)Start Code PrefixNAL Header ByteSlice HeaderAny encoded syntax elements bit data after the encoded Slice Data (PAK Object Command) of the current Slice and prior to the next encoded Slice Data of the next Slice or prior to the end of the bistream, whichever comes firstCabac_Zero_Word or Trailing_Zero_8bits (as many bytes as there is). Anything listed above before a Slice DataContext switch interrupt is not supported by this command.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td>Command Type
Default Value: 3h PARALLEL_VIDEO_PIPE
Format: OpCode</td></tr>
<tr><td></td><td>28:27</td><td>Pipeline
Default Value: 2h MFX_PAK_INSERT_OBJECT
Format: OpCode</td></tr>
<tr><td></td><td>26:24</td><td>Media Command Opcode
Default Value: 0h MFX_COMMON
Format: OpCode</td></tr>
<tr><td></td><td>23:21</td><td>SubOpcode A
Default Value: 2h
Format: OpCode</td></tr>
<tr><td></td><td>20:16</td><td>SubOpcode B
Default Value: 8h
Format: OpCode</td></tr>
<tr><td></td><td>15:12</td><td>Reserved</td></tr>
<tr><td></td><td>11:0</td><td>DWord Length
Default Value: 0h Excludes DWord (0,1) = Variable Length in DW
Format: =n Total Length - 2</td></tr>
<tr><td>1</td><td>31:18</td><td>Reserved
Format: MBZ</td></tr>
<tr><td></td><td>17:16</td><td>DataByteOffset - SrcDataStartingByteOffset[1:0]
Source Data Starting Byte Position within the very first inline DW.
Programming Notes
Must be set to 0 for JPEG encoder</td></tr>
<tr><td></td><td>15</td><td>HeaderLengthExcludeFrmSize
In case this flag is on, bits are NOT accumulated during current access unit coding neither for Cabac Zero Word insertion bits counting or for output in MMIO register MFC_BITSTREAM_BYTECOUNT_FRAME_NO_HEADER. When using HeaderLenghtExcludeFrmSize for header insertion, the software needs to make sure that data comes already with inserted start code emulation bytes. SW shouldn't set EmulationFlag bit (Bit 3 of DWORD1 of MFX_PAK_INSERT_OBJECT).
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>1</td><td>NO_ACCUMULATION</td><td>Bits during current call are not accumulated</td></tr>
<tr><td>0</td><td>ACCUMULATE</td><td>All bits accumulated</td></tr>
</table>
<table>
<tr><th>Programming Notes</th><th>Project</th></tr>
<tr><td>Must be set to 0 for JPEG encoder</td><td>CHV, BSW</td></tr>
</table>
</td></tr>
</table>

MFX_PAK_INSERT_OBJECT

	14	**Slice Header Indicator** This bit indicates if the insert object is a slice header. In the VDEnc mode, PAK only gets this command at the beginning of the frame for slice position X=0, Y=0. It internally generates the header that needs to be inserted per slice. For VDEnc mode, this bit should always be set. **Value** | **Name** | **Description** 1 | SLICE_HEADER | Insertion Object is a Slice Header. The command is stored internally by HW and is used for inserting slice headers. 0 | LEGACY | Legacy Insertion Object command. The PAK Insertion Object command is not stored in HW. **Programming Notes** In VDENC mode, we support only Slice layer without partitioning RBSP syntax. The payload for PAK_INS_OBJ should contain only start code for Slice header followed by NAL_type and slice header (slice_header() in AVC spec). The payload for PAK_INS_OBJ shouldn't contain CABAC Byte alignment bits. HW adds these alignment bits which are part of slice_data. Example PAK_INS_OBJ payload : 00 00 01 <NAL_type> <slice_header_Byte0> ....<slice_header_Byte LAST> Any zero_bytes that are added before slice header can be inserted by any preceding general PAK_INS_OBJ.
	13:8	**DataBitsInLastDW - SrCDataEndingBitInclusion[5:0]** Source Data to be included in the very last inline DW. Follows the MSBit is the upper bit of each byte within the DW. The lower byte is actually processed first.For example, SrCDataEndingBitInclusion = 9, bit 7:0 and bit 15 are included as valid header data. **Value** | **Name** [1,32] |
	7:4	**SkipEmulByteCnt - Skip Emulation Byte Count** Skip emulation check for number of starting bytesIt can be programmed from 0 to 15 bytes.For example, to skip the start code that has already prefixed in the bitstream. **Programming Notes** Must be set to 0 for JPEG encoder
	3	**EmulationFlag - EmulationByteBitsInsertEnable** **Value** | **Name** | **Description** 0 | NONE | No emulation 1 | EMULATE | Instruct the hardware to perform Start Code Prefix (0x 00 00 01/02/03/00) Search and Prevention Byte (0x 03) insertion on the insertion data of this command. It is required that hardware will handle a start code prefix crossing the boundary between insertion commands, or an insertion command followed by a PAK Object command. **Programming Notes** Must be set to 0 for JPEG encoder

<table>
<tr><th colspan="3">MFX_PAK_INSERT_OBJECT</th></tr>
<tr><td></td><td>2</td><td>LastHeaderFlag - LastSrcHeaderDataInsertCommandFlag
To process a series of consecutive insertion commands, this flag (=1) indicates the current command is the last 'header' insertion in the series.In CABAC, hardware must perform the "1" insert for byte align for Slice Header before Slice Data comes in in the next PAK-OBJECT command.In CAVLC, hardware ignores this bit</td></tr>
<tr><td></td><td>1</td><td>EndOfSliceFlag - LastDstDataInsertCommandFlag
No more insertion command and no more PAK-OBJECT command follows.Flush data out to memory</td></tr>
<tr><td></td><td>0</td><td>BitstreamStartReset - ResetBitStreamStartingPos
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>1</td><td>RESET</td><td>Reset the bitstream buffer insertion position to the bitstream buffer starting position.</td></tr>
<tr><td>0</td><td>INSERT</td><td>Insert the current command inline data starting at the current bitstream buffer insertion position</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>Must be set to 1 for JPEG encoder</td></tr>
</table>
</td></tr>
<tr><td>2..n</td><td>31:0</td><td>Insert Data PayLoad
Actual Data to be inserted to the output bitstream buffer.</td></tr>
</table>

MFX_PIPE_BUF_ADDR_STATE

MFX_PIPE_BUF_ADDR_STATE

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_PIPE_BUF_ADDR_STATE Format: OpCode
	26:24	**Common Opcode** Default Value: 0h MFX_COMMON_STATE Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 2h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Fixed Length

Value	Name	Description	Project
3Bh	DWORD_COUNT_n **[Default]**	Excludes DWord (0,1)	CHV, BSW

MFX_PIPE_BUF_ADDR_STATE

DWord	Bit	Description
1	31:6	**Pre Deblocking Destination Address** Format: GraphicsAddress[31:6] Specifies the 4K byte aligned frame buffer address for outputting the non-filtered reconstructed YUV picture (i.e. output of final adder in each codec standard, and prior to the deblocking filter unit). This field is ignored if PreDeblockOutEnable is set to 0 (disable).
	5:0	**Reserved** Project: CHV, BSW
2	31:16	**Reserved** Format: MBZ
	15:0	**Pre Deblocking Destination Address High** Project: CHV, BSW Format: GraphicsAddress[47:32] This field is for the upper range of Pre-Deblocking Destination Address. This field is ignored if **PreDeblockOutEnable** is set to 0 (disable).
3	31:15	**Reserved** Format: MBZ
	14:13	**Reserved**
	12:11	**Reserved** Format: MBZ
	10:9	**Reserved** Project: CHV, BSW Format: MBZ
	8:7	**Pre Deblocking - Arbitration Priority Control** Project: CHV, BSW 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
	6:5	**Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for Pre Deblocking Destination Address** Project: CHV, BSW This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream.

MFX_PIPE_BUF_ADDR_STATE

Value	Name	Description
00b	Use Cacheability Controls from page table	
01b	UC	Uncacheable - non-cacheable
10b	WT	Writethrough
11b	WB	Writeback

4:3 **Target Cache (TC) Pre Deblocking Destination Address**

Project:	CHV, BSW

This field allows the choice of LLC vs eLLC for caching.

Value	Name
00b	eLLC Only - not snooped in GT
01b	LLC Only
10b	LLC/eLLC Allowed
11b	L3, LLC, eLLC Allowed

2 **Encrypted Data Pre Deblocking Destination Address**

Project:	CHV, BSW
Format:	Enable

This field controls whether data is decrypted while being read. This field is ignored for writes.

1:0 **Age for QUADLRU (AGE) Pre Deblocking Destination Address**

Project:	CHV, BSW

This field allows the selection of AGE parameter for a given surface in LLC or eLLC. If a particular allocation is done at youngest age ("3") it tends to stay longer in the cache as compared to older age allocations ("2", "1", or "0"). This option is given to driver to be able to decide which surfaces are more likely to generate HITs, hence need to be replaced least often in caches.

Value	Name
11b	Good chance of generating hits
10b	Next good chance of generating hits
01b	Decent chance of generating hits
00b	Poor chance of generating hits

4 | 31:6 **Post Deblocking Destination Address**

Format:	GraphicsAddress[31:6]

Specifies the 4K byte aligned frame buffer address for outputting the post-loop filtered reconstructed YUV picture (i.e. output of the deblocking filter unit)This field is ignored if PostDeblockOutEnable is set to 0 (disable).

Programming Notes
In SVC mode, this surface is used as following: (1) Decoder Mode (a) QBL or Final Layer - this is for filtered reconstructed YUV picture (b) Next Layer is quality layer - this is for filtered Interlayer coefpred surface (c) Next Layer is enhancement layer - this is for filtered Interlayer

MFX_PIPE_BUF_ADDR_STATE

DWord	Bit	Description
		intra surface (2) Encoder Mode (a) Last Layer - this is for filtered reconstructed YUV picture (b) Next Layer is quality layer - this is for filtered Interlayer coefpred surface (c) Next Layer is enhancement layer - this is for filtered Interlayer intra surface or filtered reconstructed YUV picture
	5:0	**Reserved** Project: CHV, BSW
5	31:16	**Reserved** Format: MBZ
	15:0	**Post Deblocking Destination Address High** Project: CHV, BSW Format: GraphicsAddress[47:32] This field is for the upper range of Post-Deblocking Destination Address. This field is ignored if **PostDeblockOutEnable** is set to 0 (disable).
6	31:15	**Reserved** Format: MBZ
	14:13	**Reserved** Project: CHV, BSW Format: MBZ
	12:11	**Reserved** Format: MBZ
	10:9	**Reserved** Project: CHV, BSW Format: MBZ
	8:7	**Post Deblocking - Arbitration Priority Control** Project: CHV, BSW 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
	6:5	**Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for Post Deblocking Destination Address** Project: CHV, BSW This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream.

MFX_PIPE_BUF_ADDR_STATE

Value	Name	Description
00b	Use Cacheability Controls from page table	
01b	UC	Uncacheable - non-cacheable
10b	WT	Writethrough
11b	WB	Writeback

4:3 **Target Cache (TC) for Post Deblocking Destination Address**

Project:	CHV, BSW

This field controls the L3$, LLC and eLLC (eDRAM) cacheability for a given surface. Setting of "00" points to PTE settings which defaults to eDRAM (when present). If no eDRAM, the access will be allocated to LLC. Setting of "01", allocates into LLC and victimizes the line to eDRAM. Setting of "10" allows the line to be allocated in either LLC or eDRAM. Setting of "11" is the only option for a memory access to be allocated in L3$ as well as LLC/eLLC

00b: eLLC Only *("00" setting points TC selection to PTE which defaults to eLLC)*

01b: LLC Only (*Works at the allocation time, later victimization from LLC downgrades the line to eLLC if present*).

10b: LLC/eLLC Allowed.

11b: L3, LLC, eLLC Allowed.

Errata CHV:A-E (FIXED BY:G0 Stepping):

For all system that does NOT use SVM (i.e. coherent L3$ surfaces), back snoops from LLC has to be disabled (**Dis_GtCvUpdtOnRd = “1”**). Than target Cache settings can be programmed as POR requirements of L3/LLC/eDRAM caching.

For all systems that does use SVM (i.e. coherent L3$ surfaces), the recomended setting would be "00" in target cache settings. In case of L3 surfaces, the performance has to be tuned between "00" and "11" setting based on the benefits of L3 caching outweighting the degradation of backsnoops.

Post G0-stepping, the above w/a for coherent L3$ surfaces is not needed.

Value	Name
00b	eLLC Only - not snooped in GT
01b	LLC Only
10b	LLC/eLLC Allowed
11b	L3, LLC, eLLC Allowed

2 **Reserved**

Project:	CHV, BSW

1:0 **Age for QUADLRU (AGE) for Post Deblocking Destination Address**

Project:	CHV, BSW

This field allows the selection of AGE parameter for a given surface in LLC or eLLC. . If a particular allocation is done at youngest age ("3") it tends to stay longer in the cache as compared to older age allocations ("2", "1", or "0"). This option is given to driver to be able to decide which surfaces are more likely to generate HITs, hence need to be replaced least often in caches.

MFX_PIPE_BUF_ADDR_STATE

DWord	Bit	Description
		Value / **Name**: 11b – Good chance of generating hits.; 10b – Next good chance of generating hits; 01b – Decent good chance of generating hits; 00b – Poor good chance of generating hits
7	31:6	**Original Uncompressed Picture Source Address** Format: GraphicsAddress[31:6] Specifies the 64 byte aligned frame buffer address for fetching YUV pixel data from the original uncompressed input picture for encoding. This field is only valid in **encoding** mode.
	5:0	**Reserved** Project: CHV, BSW Format: MBZ
8	31:16	**Reserved** Format: MBZ
	15:0	**Original Uncompressed Picture Source Address High** Project: CHV, BSW Format: GraphicsAddress[47:32] This field is for the upper range of Original Uncompressed Picture Source Address. This field is valid for **encoding** mode only.
9	31:15	**Reserved** Format: MBZ
	14:13	**Reserved** Project: CHV, BSW Format: MBZ
	12:11	**Reserved** Format: MBZ
	8:7	**Original Uncompressed Picture Source - Arbitration Priority Control** Project: CHV, BSW 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

MFX_PIPE_BUF_ADDR_STATE

DWord	Bit	Description
	6:5	**Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for Original Uncompressed Picture Source Address**

Project:	CHV, BSW

This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream.

Value	Name	Description
00b	Use Cacheability Controls from page table	
01b	UC	Uncacheable - non-cacheable
10b	WT	Writethrough
11b	WB	Writeback

DWord	Bit	Description
	4:3	**Target Cache (TC) for Original Uncompressed Picture Source Address**

Project:	CHV, BSW

This field allows the choice of LLC vs eLLC for caching

Value	Name
00b	eLLC Only - not snooped in GT
01b	LLC Only
10b	LLC/eLLC Allowed
11b	L3, LLC, eLLC Allowed

DWord	Bit	Description
	2	**Reserved**
	1:0	**Age for QUADLRU (AGE) for Original Uncompressed Picture Source Address**

Project:	CHV, BSW

This field allows the selection of AGE parameter for a given surface in LLC or eLLC. . If a particular allocation is done at youngest age ("3") it tends to stay longer in the cache as compared to older age allocations ("2", "1", or "0"). This option is given to driver to be able to decide which surfaces are more likely to generate HITs, hence need to be replaced least often in caches.

Value	Name
11b	Good chance of generating hits.
10b	Next good chance of generating hits
01b	Decent chance of generating hits
00b	Poor chance of generating hits

DWord	Bit	Description
10	31:6	**StreamOut Data Destination Base Address**

Format:	GraphicsAddress[31:6]

Specifies the 64 byte aligned address for outputting the per-MB indirect data to memory when **StreamOutEnable** is set in the MFX_PIPE_MODE_SELECT command. For Decoder : This field is used for transcoding purpose. For Encoder : This field is used for dynamic repeat of frame in PAK for Rate Control. Also used for feeding coding information back to the Host, Video Preprocessing Unit and ENC Unit. All data are written in fixed formats, and therefore all record sizes are known in the hardware. Hardware can calculate the offset into this base address for per-MB data.

MFX_PIPE_BUF_ADDR_STATE		
	5:0	**Reserved** Project: CHV, BSW Format: MBZ
11	31:16	**Reserved** Format: MBZ
	15:0	**StreamOut Data Destination Base Address High** Project: CHV, BSW Format: GraphicsAddress[47:32] This field is for the upper range of Original Uncompressed Picture Source Address
12	31:15	**Reserved** Format: MBZ
	14:13	**Reserved** Project: CHV, BSW Format: MBZ
	12:11	**Reserved** Format: MBZ
	8:7	**StreamOut Data Destination - Arbitration Priority Control** Project: CHV, BSW 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
	6:5	**Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for StreamOut Data Destination Base Address** Project: CHV, BSW This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream. **Value** | **Name** | **Description** 00b | Use Cacheability Controls from page table | 01b | UC | Uncacheable - non-cacheable 10b | WT | Writethrough 11b | WB | Writeback

MFX_PIPE_BUF_ADDR_STATE

	4:3	**Target Cache (TC) for StreamOut Data Destination Base Address** Project: CHV, BSW This field allows the choice of LLC vs eLLC for caching **Value** / **Name** 00b: eLLC Only - not snooped in GT 01b: LLC Only 10b: LLC/eLLC Allowed 11b: L3, LLC, eLLC Allowed
	2	**Reserved** Project: CHV, BSW
	1:0	**Age for QUADLRU (AGE) for StreamOut Data Destination Base Address** Project: CHV, BSW This field allows the selection of AGE parameter for a given surface in LLC or eLLC. If a particular allocation is done at youngest age ("3") it tends to stay longer in the cache as compared to older age allocations ("2", "1", or "0"). This option is given to driver to be able to decide which surfaces are more likely to generate HITs, hence need to be replaced least often in caches. **Value** / **Name** 11b: Good chance of generating hits. 10b: Next good chance of generating hits 01b: Decent chance of generating hits 00b: Poor chance of generating hits
13	31:6	**Intra Row Store Scratch Buffer Base Address** Format: GraphicsAddress[31:6] This field provides the base address of the scratch buffer (read/write) used by the AVC/SVC/VP8 IntraPrediction unit to store MB information of the previous row for processing of each macroblock in the current row. The Intra Row Store buffer must be 64-byte cacheline aligned.Hardware uses the horizontal address of the current macroblock to address the Intra Row Store. This field is ignored in MPEG2 and VC1 mode. Max 256 cachelines for 4K pixels (1 cacheline for either MBAFF or non-MBAFF)Intra Row Store Scratch Buffer - Arbitration Priority Control **Programming Notes** This is one of the four RowStore Scratch Buffers which can be programmed to use the internal Media Cache (total size 640 CacheLine). When Intra Row Store Scratch Buffer Cache Select is programmed to "1", this data will be stored inside MFX Media Internal Stroage. Driver then needs to program this Base Address between 0 to 639, indicating starting cachelines address to Media Cache. Driver needs to make sure the whole buffer fits into MFX Media Internal Storage. *(Notes: 1 cacheline per MB, and the buffer needs to have enough space for 1 MB row).*
	5:0	**Reserved**
14	31:16	**Reserved** Format: MBZ

MFX_PIPE_BUF_ADDR_STATE

DWord	Bit	Description
	15:0	**Intra Row Store Scratch Buffer Base Address High** Project: CHV, BSW Format: GraphicsAddress[47:32] This field is for the upper range of Intra RowStore/Scratch Buffer Base Address This field is ignored in MPEG2 and VC1 mode.
15	31:15	**Reserved** Format: MBZ
	14:13	**Reserved** Project: CHV, BSW Format: MBZ
	12	**Intra Row Store Scratch Buffer Cache Select** Project: CHV, BSW This field controls if Intra Row Store is going to store inside Media Cache or to LLC. Value 0: Buffer going to LLC. Value 1: Buffer going to Internal Media Storage
	11	**Reserved** Project: CHV, BSW Format: MBZ
	10:9	**Reserved** Project: CHV, BSW Format: MBZ
	8:7	**Intra Row Store Scratch Buffer - Arbitration Priority Control** Project: CHV, BSW This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. 00b: Highest priority 01b: Second highest priority 10b: Third highest priority 11b: Lowest priority
	6:5	**Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for Intra Row Store Scratch Buffer Base Address** Project: CHV, BSW This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream.

Nested tables for bit 12:

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

Nested table for bits 8:7:

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

MFX_PIPE_BUF_ADDR_STATE

		<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>00b</td><td>Use Cacheability Controls from page table</td><td></td></tr><tr><td>01b</td><td>UC</td><td>Uncacheable - non-cacheable</td></tr><tr><td>10b</td><td>WT</td><td>Writethrough</td></tr><tr><td>11b</td><td>WB</td><td>Writeback</td></tr></table>
	4:3	**Reserved**
	2	**Reserved**
	1:0	**Age for QUADLRU (AGE) for Intra Row Store Scratch Buffer Base Address** <table><tr><td>Project:</td><td>CHV, BSW</td></tr></table>This field allows the selection of AGE parameter for a given surface in LLC or eLLC. If a particular allocation is done at youngest age ("3") it tends to stay longer in the cache as compared to older age allocations ("2", "1", or "0"). This option is given to driver to be able to decide which surfaces are more likely to generate HITs, hence need to be replaced least often in caches.<table><tr><th>Value</th><th>Name</th></tr><tr><td>11b</td><td>Good chance of generating hits.</td></tr><tr><td>10b</td><td>Next good chance of generating hits</td></tr><tr><td>01b</td><td>Decent chance of generating hits</td></tr><tr><td>00b</td><td>Poor chance of generating hits</td></tr></table>
16	31:6	**Deblocking Filter Row Store Scratch Base Address** <table><tr><td>Format:</td><td>GraphicsAddress[31:6]</td></tr></table>Deblocking Filter Row Store is needed for: • AVC and VC1 In-Loop Deblocking Filter • VC1 Overlap-smoothing Filter • AVC, VC1, and MPEG-2 Out-Of-Loop Deblocking Filter (Intel extension) This field provides the 64 byte aligned base address of the scratch buffer (read and write) used by the deblocking filter unit to store MB information of the previous row for filtering of each macroblock in the current row. The Deblocking Filter Row Store buffer must be 64-byte cacheline aligned. Hardware uses the horizontal address of the current macroblock to address the Deblocking Filter Row Store. Max 6 cachelines for VC1 and MPEG2, and max 4 for AVC (for MBAFF, 2 for non-MBAFF)<table><tr><th>Programming Notes</th></tr><tr><td>This is one of the four RowStore Scratch Buffers which can programmed to use the internal Media Cache (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 cachelines address location for this buffer. Driver needs to make sure the whole buffer fits into Media Internal Storage. (Notes: 2 cachelines per MB for non-mbaff; 4 cahcelines per MB pair for mbaff, and the buffer needs to have enough space for 1 MB (pair) row).</td></tr></table>

MFX_PIPE_BUF_ADDR_STATE

DWord	Bit	Description
	5:0	**Reserved**
17	31:16	**Reserved**
	15:0	**Deblocking Filter Row Store Scratch Base Address High** Project: CHV, BSW Format: GraphicsAddress[47:32] This field is for the upper range of Deblocking Filter Row Store Scratch Buffer Address.
18	31:15	**Reserved**
	14:13	**Reserved** Project: CHV, BSW Format: MBZ
	12	**Deblocking Filter Row Store Scratch Buffer Cache Select** Project: CHV, BSW 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 Media Internal Storage

DWord	Bit	Description
	11	**Reserved**
	10:9	**Reserved**
	8:7	**Deblocking Filter Row Store Scratch - Arbitration Priority Control** Project: CHV, BSW 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

DWord	Bit	Description
	6:5	**Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for Deblocking Filter Row Store Scratch Base Address** Project: CHV, BSW This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream.

Value	Name	Description
00b	Use Cacheability Controls from page table	
01b	UC	Uncacheable - non-cacheable
10b	WT	Writethrough
11b	WB	Writeback

MFX_PIPE_BUF_ADDR_STATE

	4:3	**Target Cache (TC) for Deblocking Filter Row Store Scratch Base Address** Project: CHV, BSW This field allows the choice of LLC vs eLLC for caching **Value** / **Name** 00b: eLLC Only - not snooped in GT 01b: LLC Only 10b: LLC/eLLC Allowed 11b: L3, LLC, eLLC Allowed
	2	**Reserved**
	1:0	**Age for QUADLRU (AGE) for Deblocking Filter Row Store Scratch Base Address** This field allows the selection of AGE parameter for a given surface in LLC or eLLC.If a particular allocation is done at youngest age ("3") it tends to stay longer in the cache as compared to older age allocations ("2", "1", or "0"). This option is given to driver to be able to decide which surfaces are more likely to generate HITs, hence need to be replaced least often in caches. **Value** / **Name** 11b: Good chance of generating hits. 10b: Next good chance of generating hits 01b: Decent chance of generating hits 00b: Poor chance of generating hits
19..50	63:48	**Reserved**
	47:32	**Reference Picture Address [n] High** Project: CHV, BSW Format: Address[47:32] This field is for the upper range of Reference Picture Addresses
	31:6	**Reference Picture Address [n]** Format: Address[31:6] Specifies the 64 byte aligned reference frame buffer addresses for the motion compensation operation in AVC/ /MPEG2. AVC can specify up to 16 YUV frame-based surfaces for both forward and backward references, i.e. L0+L1 total = 16 max. Any entry can be assigned to L0 or L1 or both lists.But VC1 and MPEG2, worst case, can use up to 2 YUV frame-based surfaces for both forward and backward references: • P-MB : RefAddr[0] - temporal closest previous field of a reference frame (can be the current frame) • RefAddr[1]- next temporal closest previous field of a reference frame (must be different from the current frame) It is a variant (without the LongTermRefPic specification) of the RefFrameList[16] defined in AVC DXVA Spec. RefAddr[0-15] is indexed by frame_storeID »1. It is not a packed list, i.e. invalid entries can scatter among the list. All invalid addresses must be set to a valid address RefAddr[0] by the driver. The same applies to VC1 and MPEG2.

MFX_PIPE_BUF_ADDR_STATE

DWord	Bit	Description
		Programming Notes AVC: Always specifies all 16 addresses even some of them are not needed as indicated by the max num of active reference pictures. This is done for preventing data corruption (error, fault condition, etc.) by having all the references being set to a legal location.
	5:0	**Reserved** Project: CHV, BSW Format: MBZ
51	31:15	**Reserved** Format: MBZ
	14:13	**Reserved** Project: CHV, BSW Format: MBZ
	12:9	**Reserved** Format: MBZ
	8:7	**Reference Picture - Arbitration Priority Control** Project: CHV, BSW 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
	6:5	**Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for Reference Picture Addresses** Project: CHV, BSW This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream. **Note: There is ONLY ONE LLC/eLLC Cacheability Control (LeLLCCC) for all 16 Reference Picture Addresses (RefAddr[0-15])** Value / Name / Description: 00b: Use Cacheability Controls from page table 01b: UC — Uncacheable - non-cacheable 10b: WT — Writethrough 11b: WB — Writeback
	4:3	**Target Cache (TC) for Reference Picture Addresses** Project: CHV, BSW This field allows the choice of LLC vs eLLC for caching. **NOTE: There is ONLY ONE Target Cache (TC) for all 16 Reference Picture Addresses (RefAddr[0-15])**

MFX_PIPE_BUF_ADDR_STATE

DWord	Bit	Description
		<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>eLLC Only - not snooped in GT</td></tr><tr><td>01b</td><td>LLC Only</td></tr><tr><td>10b</td><td>LLC/eLLC Allowed</td></tr><tr><td>11b</td><td>L3, LLC, eLLC Allowed</td></tr></table>
	2	**Reserved** <table><tr><td>Project:</td><td>CHV, BSW</td></tr></table>
	1:0	**Age for QUADLRU (AGE) for Reference Picture Addresses** <table><tr><td>Project:</td><td>CHV, BSW</td></tr></table>This field allows the selection of AGE parameter for a given surface in LLC or eLLC.If a particular allocation is done at youngest age ("3") it tends to stay longer in the cache as compared to older age allocations ("2", "1", or "0"). This option is given to driver to be able to decide which surfaces are more likely to generate HITs, hence need to be replaced least often in caches. **NOTE: There is ONLY ONE Age for QUADLRU (AGE) for all 16 Reference Picture Addresses (RefAddr[0-15])** <table><tr><th>Value</th><th>Name</th></tr><tr><td>11b</td><td>Good chance of generating hits.</td></tr><tr><td>10b</td><td>Next good chance of generating hits</td></tr><tr><td>01b</td><td>Decent chance of generating hits</td></tr><tr><td>00b</td><td>Poor chance of generating hits</td></tr></table>
52	31:6	**Macroblock Buffer Base Address or Decoded Picture Error/Status Buffer Base Address** <table><tr><td>Project:</td><td>CHV, BSW)</td></tr><tr><td>Format:</td><td>GraphicsAddress[31:6]</td></tr></table>**For decoder:** Specifies the 64 byte aligned buffer address for writing a single error/status record into the memory when **Pic Error/Status Report Enable** is set in the MFX_PIPE_MODE_SELECT Command. The error/status record is written by HW at the end of decoding one single picture. The content of this memory location is not encrypted and can be later read by the driver only. The record is written in a fixed format, total 96-bits in size always. Please refer to "Media VDBOX -> Video Codec -> Other Codec Functions -> MFX Error Handling -> Decoder" session for the output format. **For encoder:** Specifies the 64 byte aligned buffer address for reading the per-MB indirect data from memory when **MacroblockStatEnable** is set in the MFX_AVC_IMG_STATE Command. This field is used for dynamic repeat of frame in PAK for Rate Control. Also used for feeding coding information back to the Host, Video Preprocessing Unit, and ENC Unit. All data are written in fixed formats, and therefore all record sizes are known in the hardware. Hardware can calculate the offset into this base address for per-MB data.
	5:0	**Reserved** <table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table>

MFX_PIPE_BUF_ADDR_STATE

DWord	Bit	Description
53	31:16	**Reserved** Format: MBZ
	15:0	**Macroblock Buffer Base Address or Decoded Picture Error/Status Buffer Base Address High** Project: CHV, BSW Format: GraphicsAddress[47:32] This field is for the upper range of Macroblock Status Buffer Base Address
54	31:15	**Reserved** Format: MBZ
	14:13	**Reserved** Project: CHV, BSW Format: MBZ
	12:11	**Reserved** Format: MBZ
	10:9	**Reserved** Project: CHV, BSW Format: MBZ
	8:7	**Macroblock Status Buffer - Arbitration Priority Control** Project: CHV, BSW 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
	6:5	**Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for Macroblock Status Buffer Base Address** Project: CHV, BSW This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream. Value / Name / Description: 00b: Use Cacheability Controls from page table 01b: UC — Uncacheable - non-cacheable 10b: WT — Writethrough 11b: WB — Writeback

MFX_PIPE_BUF_ADDR_STATE

DWord	Bit	Description
	4:3	**Target Cache (TC) for Macroblock Status Buffer Base Address** Project: CHV, BSW This field allows the choice of LLC vs eLLC for caching. **Value** / **Name** 00b: eLLC Only - not snooped in GT 01b: LLC Only 10b: LLC/eLLC Allowed 11b: L3, LLC, eLLC Allowed
	2	**Reserved** Project: CHV, BSW
	1:0	**Age for QUADLRU (AGE) for Macroblock Status Buffer Base Address** Project: CHV, BSW This field allows the selection of AGE parameter for a given surface in LLC or eLLC.If a particular allocation is done at youngest age ("3") it tends to stay longer in the cache as compared to older age allocations ("2", "1", or "0"). This option is given to driver to be able to decide which surfaces are more likely to generate HITs, hence need to be replaced least often in caches. **Value** / **Name** 11b: Good chance of generating hits. 10b: Next good chance of generating hits 01b: Decent chance of generating hits 00b: Poor chance of generating hits
55	31:6	**Macroblock ILDB StreamOut Buffer Base Address** Format: GraphicsAddress[31:6] Specifies the 64 byte aligned buffer address for writing MB ILDB parameter per MB to memory when **Debocker streamout enable** is set in the MFX_PIPE_MODE_SELECT Command. The ildb MB control parameters are written by HW at the end of each decoding MB. Only AVC edge information is being streamed out. It is used in AVC decode mode only.
	5:0	**Reserved** Project: CHV, BSW Format: MBZ
56	31:16	**Reserved** Format: MBZ
	15:0	**Macroblock ILDB StreamOut Buffer Base Address High** Project: CHV, BSW Format: GraphicsAddress[47:32] This field is for the upper range of Deblocking Filter Row Store Scratch Address

MFX_PIPE_BUF_ADDR_STATE

DWord	Bit	Description
57	31:15	**Reserved** Format: MBZ
	14:13	**Reserved** Project: CHV, BSW Format: MBZ
	12:11	**Reserved** Format: MBZ
	10:9	**Reserved** Project: CHV, BSW Format: MBZ
	8:7	**Macroblock ILDB StreamOut Buffer - Arbitration Priority Control** Project: CHV, BSW 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
	6:5	**Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for Macroblock ILDB StreamOut Buffer Base Address** Project: CHV, BSW This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream. Value / Name / Description: 00b: Use Cacheability Controls from page table 01b: UC — Uncacheable - non-cacheable 10b: WT — Writethrough 11b: WB — Writeback
	4:3	**Target Cache (TC) for Macroblock ILDB StreamOut Buffer Base Address** Project: CHV, BSW This field allows the choice of LLC vs eLLC for caching. Value / Name: 00b: eLLC Only - not snooped in GT 01b: LLC Only 10b: LLC/eLLC Allowed 11b: L3, LLC, eLLC Allowed

MFX_PIPE_BUF_ADDR_STATE

	2	**Encrypted Data for Macroblock ILDB StreamOut Buffer Base Address** Project: CHV, BSW Format: Enable This field controls whether data is decrypted while being read. This field is ignored for writes.
	1:0	**Age for QUADLRU (AGE) for Macroblock ILDB StreamOut Buffer Base Address** Project: CHV, BSW This field allows the selection of AGE parameter for a given surface in LLC or eLLC.If a particular allocation is done at youngest age ("3") it tends to stay longer in the cache as compared to older age allocations ("2", "1", or "0"). This option is given to driver to be able to decide which surfaces are more likely to generate HITs, hence need to be replaced least often in caches. Value / Name: 11b: Good chance of generating hits. 10b: Next good chance of generating hits 01b: Decent chance of generating hits 00b: Poor chance of generating hits
58	31:6	**Second Macroblock ILDB StreamOut Buffer Base Address** Format: GraphicsAddress[31:6] 64 byte aligned buffer. Specifies the 64 byte aligned buffer address for writing MB ILDB parameter per MB to memory when Debocker streamout enable is set in the MFX_PIPE_MODE_SELECT Command. The ildb MB control parameters are written by HW at the end of each decoding MB. Only AVC edge information is being streamed out. It is used in AVC decode mode only.
	5:0	**Reserved** Project: CHV, BSW Format: MBZ
59	31:16	**Reserved** Format: MBZ
	15:0	**Second Macroblock ILDB StreamOut Buffer Base Address High** Project: CHV, BSW Format: GraphicsAddress[47:32] This field is for the upper range of Second Macroblock ILDB StreamOutBuffer Base Address.
60	31:15	**Reserved** Format: MBZ
	14:13	**Reserved** Project: CHV, BSW Format: MBZ

MFX_PIPE_BUF_ADDR_STATE

	12:11	**Reserved** Format: MBZ
	10:9	**Reserved** Project: CHV, BSW Format: MBZ
	8:7	**Second Macroblock ILDB StreamOut Buffer - Arbitration Priority Control** Project: CHV, BSW This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.
	6:5	**Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for Second Macroblock ILDB StreamOut Buffer Base Address** Project: CHV, BSW This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream.

Value	Name	Description
00b	Use Cacheability Controls from page table	
01b	UC	Uncacheable
10b	WT	Writethrough
11b	WB	Writeback

	4:3	**Second Macroblock ILDB StreamOut Buffer Base Address - Target Cache (TC)** Project: CHV, BSW This field allows the choice of LLC vs eLLC for caching

Value	Name	Description
00b	eLLC Only	not snooped in GT
01b	LLC Only	
10b	LLC/eLLC Allowed	
11b	L3, LLC, eLLC Allowed	

	2	**Second Macroblock ILDB StreamOut Buffer Base Address - Encrypted Data** Project: CHV, BSW Format: Enable This field controls whether data is decrypted while being read. This field is ignored for writes.
	1:0	**Age for QUADLRU (AGE) for Second Macroblock ILDB StreamOut Buffer Base Address** Project: CHV, BSW This field allows the selection of AGE parameter for a given surface in LLC or eLLC. . If a particular allocation is done at youngest age ("3") it tends to stay longer in the cache as compared to older age allocations ("2", "1", or "0"). This option is given to driver to be able to decide which surfaces are more likely to generate HITs, hence need to be replaced least often in caches.

MFX_PIPE_BUF_ADDR_STATE			
		Value	**Name**
		00b	Poor chance of generating hits
		01b	Decent chance of generating hits
		10b	Next good chance of generating hits
		11b	Good chance of generating hits

MFX_PIPE_MODE_SELECT

MFX_PIPE_MODE_SELECT		
Project:	CHV, BSW	
Source:	VideoCS	
Length Bias:	2	

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

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

<table>
<tr><th colspan="3">MFX_PIPE_MODE_SELECT</th></tr>
<tr><td></td><td>29</td><td>Reserved</td></tr>
<tr><td></td><td>28:27</td><td>Reserved</td></tr>
<tr><td></td><td>26</td><td>Reserved</td></tr>
<tr><td></td><td>25</td><td>Reserved</td></tr>
<tr><td></td><td>24</td><td>Reserved</td></tr>
<tr><td></td><td>23:19</td><td>Reserved</td></tr>
<tr><td></td><td>18</td><td>Extended stream out enable
<table><tr><td>Format:</td><td>U1</td></tr></table>
This bit can be set only when VDEnc_Mode is set.
When this bit is set and MB stream out is enabled, per MB 1CL of data is streamed out. The actual contents of the stream out are listed in Media VDBOX > Encoder VDEnc mode StreamOut Data Structure Definition.
When this bit is not set, per MB ¼ CL data is streamed out. The actual contents of the stream out are listed in Media VDBOX > Encoder StreamOut Mode Data Structure Definition.</td></tr>
<tr><td></td><td>17</td><td>Decoder Short Format Mode
For IT mode, this bit must be 0.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>1</td><td>Long Format Driver Interface</td><td>[CHV, BSW] AVC/VC1/MVC/SVC/VP8 Long Format Mode is in use.</td></tr>
<tr><td>0</td><td>Short Format Driver Interface [Default]</td><td>AVC/VC1/MVC/SVC/VP8 Short Format Mode is in use
Note: There is no Short Format for SVC and VP8 yet, so this field must be set to 1 for SVC and VP8.</td></tr>
</table></td></tr>
<tr><td></td><td>16:15</td><td>Decoder Mode select
Each coding standard supports two entry points: VLD entry point and IT (IDCT) entry point. This field selects which one is in use.This field is only valid if Codec Select is 0 (decoder).
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>VLD Mode</td><td>All codec minimum must support this mode
Configure the MFD Engine for VLD Mode Note: All codec minimum must support this mode</td></tr>
<tr><td>1h</td><td>IT Mode</td><td>Configure the MFD Engine for IT Mode Note: Only VC1 and MPEG2 support this mode</td></tr>
<tr><td>2h</td><td>Deblocker Mode</td><td>Configure the MFD Engine for Standalone Deblocker Mode. Require streamout AVC edge control information from preceeding decoding pass.</td></tr>
<tr><td>3h</td><td>Interlayer Mode</td><td>Configure the MFX Engine for standalone SVC interlayer upsampling for motion info, residual and reconstructed pixel. Require information being streamout from the preceding encoding and decoding pass of a reference layer.></td></tr>
</table></td></tr>
<tr><td></td><td>14:13</td><td>Reserved</td></tr>
</table>

MFX_PIPE_MODE_SELECT

	12	**Deblocker Stream-Out Enable** Project: CHV, BSW This field indicates if Deblocker information is going to be streamout during VLD decoding. For AVC, it is needed to enable the deblocker streamout as the AVC Disable_DLKFilterIdc is a slice level parameters. Driver needs to determine ahead of time if at least one slice of the current frame/ has deblocker ON. For SVC, there are two deblocking control streamout buffers (specified in MFX_BUF_ADDR State Command). This field is still associated with the slice level SVC Disable.DLK_Filter_Idc. **Value / Name / Description** 0h / Disable / Disable streamout of deblocking control information for standalone deblocker operation. It needs other fields to determine one or two SVC deblocking surface streamout (Post Deblocking Output Enable, Pre Deblocking Output Enable, interlayer idc and regular deblock idc). 1h / Enable /
	11	**Pic Error/Status Report Enable.** Project: CHV, BSW This field control whether the error/status reporting is enable or not.0: Disable1: EnableIn decoder modes: Error reporting is written out once per frame. The Error Report frame ID listed in DW3 along with the VLD/IT error status bits are packed into one cache and written to the "Decoded Picture Error/Status Buffer address" listed in the MFX_PIPE_BUF_ADDR_STATE Command. Note: driver shall program different error buffer addresses between pictrues; otherwise, hardware might overwrite previous written data if driver does not read it fast enough.In encoder modes: Not used Please refer to "Media VDBOX -> Video Codec -> Other Codec Functions -> MFX Error Handling -> Decoder" session for the output format. **Value / Name** 0h / Disable 1h / Enable
	10	**Stream-Out Enable** This field controls whether the macroblock parameter stream-out is enabled during VLD decoding for transcoding purpose. **Value / Name** 0h / Disable 1h / Enable **Programming Notes** In decoder modes: The Stream-Out feature is added to support transcoding. While decoding the input compressed stream, selected decoded information may be used by the encoder for re-compression.In encoder modes: This feature used to perform dynamic Multipass of PAK for conformance pupose. Also it provides feedback to host (ENC) for future needs. Software can use this bit to disable writing PAK steam data to the streamout buffer for last pass of frame in PAK. Thus, save memory bandwidth.

MFX_PIPE_MODE_SELECT

	9	**Post Deblocking Output Enable (PostDeblockOutEnable)** This field controls the output write for the reconstructed pixels AFTER the deblocking filter.In MPEG2 decoding mode, if this is enabled, VC1 deblocking filter is used. **Value** / **Name** 0h: Disable 1h: Enable
	8	**Pre Deblocking Output Enable (PreDeblockOutEnable)** This field controls the output write for the reconstructed pixels BEFORE the deblocking filter. **Value** / **Name** 0h: Disable 1h: Enable
	7:6	**Reserved** Project: CHV, BSW Format: MBZ
	5	**Stitch Mode** Exists If: //CodecSel=Encode and StandardSel=AVC **Value** / **Name** / **Description** 0h: Not in stitch mode 1h: In the special stitch mode — This mode can be used for any Codec as long as bitfield conditions are met.
	4	**Codec Select** **Value** / **Name** / **Description** 0h: Decode 1h: Encode — Valid only if StandardSel is AVC, MPEG2 and SVC)
	3:0	**Standard Select** **Value** / **Name** / **Description** 0000b: MPEG2 0001b: VC1 0010b: AVC — Covers both AVC and MVC 0011b: JPEG 0100b: SVC 0101b: VP8 — Decoder and Encoder 0110b: Reserved 0111b: Reserved 1111b: UVLD — SW decoder w/ embedded micro-controller and co-processor
2	31	**Reserved**

MFX_PIPE_MODE_SELECT

Format:	MBZ

30 Reserved

Project:	CHV, BSW

29 Reserved

Format:	MBZ

28 VMB SVC MV Replication for 8x8 Enable (Error Handling)

Project:	CHV, BSW

This bit enables Motion Vector replication on 8x8 level during SVC mode for error handling.

Value	Name	Description
0	Disable **[Default]**	Disable MV 8x8 replication in SVC mode
1	Enable	Enable MV 8x8 Replication in SVC Mode

27 VMB SVC TLB Dummy Fetch Disable for Performance

Project:	CHV, BSW

This bit disables TLB dummy fetch in SVC mode in VMB.

Value	Name	Description
0	Enable **[Default]**	Enable VMB TLB Dummy Fetch for Performance
1	Disable	Disable VMB TLB Dummy Fetch

26 Reserved

Project:	CHV, BSW

25 Reserved

Project:	CHV, BSW

24 VHR MVC Field Reference List Logic Enable

Project:	CHV, BSW

Value	Name	Description
0	Disable **[Default]**	Disable MVC Field Logic
1	Enable	VHR MVC Field Enable

23 Reserved

22:21 Reserved

20:19 Reserved

Project:	CHV, BSW
Format:	MBZ

18 Reserved

Format:	MBZ

17 Reserved

Project:	CHV, BSW

MFX_PIPE_MODE_SELECT

	16	**Reserved**
	15	**Reserved**
	14	**VLF 720i (Odd Height) in VC1 Mode** Project: CHV, BSW This bit indicates VLF write out VC1 picture with odd height (in MBs). Value / Name / Description: 0 / Disable **[Default]** / 1 / Enable / 720i Enable
	13	**Reserved**
	12	**Reserved**
	11	**Reserved**
	10	**MPC pref08x8_disable Flag (Default 0)** Value / Name: 0 / Disable 1 / Enable
	9	**Reserved** Format: MBZ
	8	**Reserved** Project: CHV, BSW
	7	**Reserved**
	6	**Clock gate Enable at Slice-level** BitFieldDesc: Value / Name / Description: 0 / Disable / Disable Slice-level Clock gating, Unit-level Clock gating will apply 1 / Enable / Enable Slice-level Clock gating, overrides any Unit level Clock gating
	5	**Reserved**
	4	**Reserved**
	3	**VDS ILDB Calculation** Project: CHV, BSW This bit forces all MB into INTRA MBs before doing ILDB control generation in VDS. Value / Name / Description: 0 / Disable **[Default]** / Use original definition for ILDB calculation. 1 / Enable / Force neighbor Intra MB = 1 on ILDB BS calculation. **Programming Notes** When the bit is '0', the ILDB control generation will be the same as the original spec (AVC/VC1/SVC).

MFX_PIPE_MODE_SELECT		
	2:1	**Reserved** Project: CHV, BSW
	0	**Reserved** Project: CHV, BSW
3	31:0	**Pic Status/Error Report ID** Exists If: //Decoder Mode Only Format: U32 In decoder modes: Error reporting is written out once per frame. This field along with the VLD error status bits are packed into one cache and written to the memory location specified by "Decoded Picture Error/Status Buffer address" listed in the MFX_PIPE_BUF_ADDR_STATE Command. Value / Name / Description: 0h / 32-bit unsigned / Unique ID Number 1h / Reserved /
4	31:0	**Media Soft-Reset Counter (per 1000 clocks)** Project: CHV, BSW In decoder modes, this indicates the number of clocks (per 1000) VINunit will wait for inactivity from MFX pipeline before issuing media soft reset. If this counter is set to 0, VINunit will never issue soft media reset. In encoder modes: This counter must be set to 0 to disable media soft reset since encoder mode is not supported. Value / Name / Project: 0 / Disable / CHV, BSW

MFX_QM_STATE

MFX_QM_STATE		
Project:	CHV, BSW	
Source:	VideoCS	
Length Bias:	2	

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

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

MFX_QM_STATE

	1:0	**AVC** Exists If: //AVC- Decoder Only **For AVC QM Type**: This field specifies which Quantizer Matrix is loaded. **Value** / **Name** 0 — AVC_4x4_Intra_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs) 1 — AVC_4x4_Inter_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs) 2 — AVC_8x8_Intra_MATRIX 3 — AVC_8x8_Inter_MATRIX
	1:0	**MPEG2** Exists If: //MPEG2- Decoder Only **For MPEG2 QM Type**: This field specifies which Quantizer Matrix is loaded. **Value** / **Name** 0 — MPEG_INTRA_QUANTIZER_MATRIX 1 — MPEG_NON_INTRA_QUANTIZER_MATRIX 2-3 — Reserved
	1:0	**JPEG** Project: CHV, BSW Exists If: //JPEG- Encoder Only **For JPEG QM Type**:This field specifies which Quantizer Matrix is loaded. **Value** / **Name** 0 — JPEG_Luma_Y_QUANTIZER_MATRIX (or R) 1 — JPEG_Chroma_Cb_QUANTIZER_MATRIX (or G) 2 — JPEG_Chroma_Cr_QUANTIZER_MATRIX (or B) **Programming Notes** For JPEG encoder, each quantization element presents 16-bit 1/QM[i][j]. In RGB encoding, because the order input image components can be RGB, GBR, BGR, YUV, the value 0 is used for the first image component, the value 1 is used for the second image component, and the value 2 is used for the third image component.
2..33	31:0	**Forward Quantizer Matrix** Project: All Format: U32 The format of a Quantizer Matrix is an 8x8 matrix in raster order. Each element is an unsigned byte.

MFX_STATE_POINTER

MFX_STATE_POINTER

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

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

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

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

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFX_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:24	**Media Command Opcode** Default Value: 0h MFX_COMMON_STATE Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 6h Format: OpCode
	15:12	**Reserved**

MFX_STATE_POINTER

	11:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Project: All Format: =n Total Length - 2
1	31:5	**State Pointer** Format: GeneralStateOffset[31:5]Indirect State Buffer Specifies the 32-byte aligned address of an Indirect State Buffer. This pointer is relative to the General State Base Address.
	4:2	**Reserved** Project: All Format: MBZ
	1:0	**State Pointer Index** Specifies one of the four indirect state pointers to program.

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

MFX_STITCH_OBJECT

MFX_STITCH_OBJECT

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

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

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

MFX_STITCH_OBJECT		
	15:14	**Reserved** Format: MBZ
	13:8	**Source Data Ending Bit Inclusion** Source Data to be included in the very last inline DW. Follows the MSBit is the upper bit of each byte within the DW. The lower byte is actually processed first.For example, SrCDataEndingBitInclusion =9, bit 7:0 and bit 15 are included as valid header data. Value: [1,32] — Name:
	7:4	**Reserved**
	3	**Reserved**
	2	**Last Source Header Data Insert Command Flag** To process a series of consecutive insertion commands, this flag (=1) indicates the current command is the last 'header' insertion in the series.In CABAC, hardware must perform the "1" insert for byte align for Slice Header before Slice Data comes in in the next PAK-OBJECT command.In CAVLC, hardware ignores this bit.
	1	**Last Destination Data Insert Command Flag** THIS FIELD MUST BE THE SAME AS Last Source Header Data Insert Command Flag No more insertion command and no more PAK-OBJECT command follows.Flush data out to memory
	0	**Reserved**
2	31:19	**Reserved** Format: MBZ
	18:0	**Indirect Data Length** Project: CHV, BSW Format: U19 This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. This field must have the same alignment as the Indirect Object Data Start Address.
3	31:0	**Indirect Data Start Address** Format: MfxIndirectBitstreamObjectAddress[31:0] This field specifies the Graphics Memory starting address of the data to be loaded into the kernel for processing. This pointer is relative to the MFX Indirect Bitstream Object Base Address. Hardware ignores this field if indirect data is not present.
4..n	31:0	**Insert Data PayLoad** Inline data to be inserted to the output bitstream buffer

MFX_SURFACE_STATE

MFX_SURFACE_STATE	
Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

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

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

Since there is only one media surface state being active during the entire encoding/decoding process, all the uncompressed/reconstructed pictures are defined to have the same surface state. For each media object call (decoding or encoding), multiple SVC surfaces can be active concurrently, to distinguish among them, a surfaceID is added to specify for each type of surface. The primary difference among picture surface states is their individual programmed base addresses, which are provided by other state commands and not included in this command. MFX engine is making the association of surface states and corresponding buffer base addresses. MFX engine currently supports only one media surface type for video and that is the NV12 (Planar YUV420 with interleaved U (Cb) and V (Cr). For optimizing memory efficiency based on access patterns, only TileY is supported. For JPEG decoder, only IMC1 and IMC3 are supported. Pitch can be wider than the Picture Width in pixels and garbage will be there at the end of each line. The following describes all the different formats that are supported and not supported in Gen7 MFX :

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

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

MFX_SURFACE_STATE

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

This surface state here is identical to the Surface State for deinterlace and sample_8x8 messages described in the Shared Function Volume and Sampler Chapter.
For non pixel data, such as row stores, indirect data (Compressed Slice Data, AVC MV record, Coeff record and AVC ILDB record) and streamin/out and output compressed bitstream, a linear buffer is employed. For row stores, the H/W is designed to guarantee legal memory accesses (read and write). For the remaining cases, indirect object base address, indirect object address upper bound, object data start address (offset) and object data length are used to fully specified their corresponding buffer. This mechanism is chosen over the pixel surface type because of their variable record sizes.
All row store surfaces are linear surface. Their addresses are programmed in Pipe_Buf_Base_State or Bsp_Buf_Base_Addr_State

Programming Notes
VC1 I picture scaling: Even though VC1 allows I reconstructed picture scaling (via RESPIC), as such scaling is only allowed at I picture. All subsequent P (and B) pictures must have the same picture dimensions with the preceding I picture. Therefore, all reference pictures for P or B picture can share the same surface state with the current P and B picture. Note : H/W is not processing RESPIC. Application is no longer expecting intel decoder pipelineand kernel to perform this function, it is going to be done in the video post-processing scaler or display controller scale as a separate step and controller.
All video codec surfaces must be NV12 Compliant, except JPEG. U/V vertical must be MB aligned for all video codec (further contrained for field picture), but JPEG can be block aligned. All video codec and JPEG uses Tiled -Y format only, for uncompressed pixel surfaces.
Even for JPEG planar 420 surface, application may provide only 1 buffers, but there is still only one single surface state for all of them. If IMC equal to 1, 2, 3 or 4, U and V have the pitch same as Y. And U and V will have different offset, each offset is block aligned.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_COMMON Format: OpCode
	26:24	**Opcode** Default Value: 0h MFX_COMMON_STATE Format: OpCode
	23:21	**SubOpA** Default Value: 0h Format: OpCode

<table>
<tr><th colspan="3">MFX_SURFACE_STATE</th></tr>
<tr><td></td><td>20:16</td><td>SubOpB<table><tr><td>Default Value:</td><td>1h</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:12</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11:0</td><td>DWord Length<table><tr><td>Format:</td><td>=n Total Length - 2</td></tr></table><table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>4h</td><td>DWORD_COUNT_n [Default]</td><td>Excludes DWord (0,1)</td></tr></table></td></tr>
<tr><td>1</td><td>31:4</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>3:0</td><td>Surface Id<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>U4</td></tr></table><table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0000b</td><td>Decoded Picture and Reference Pictures, SVC upsampling Streamout Reconstructed Pixels/Coeff_pred (Upper Layer Size)</td><td>8-bit uncompressed data</td></tr><tr><td>0001b</td><td>SVC Residual Upsampling Stream Out Surface (Upper layer Size)</td><td>16-bit uncompressed data</td></tr><tr><td>0010b</td><td>SVC Reconstructed pixel and CoeffPred Upsampling Stream In Surface (Lower Layer Size)</td><td>8-bit uncompressed data.</td></tr><tr><td>0011b</td><td>SVC Residual Upsampling Stream In Surface (lower layer size)</td><td>16-bit uncompressed data</td></tr><tr><td>0100b</td><td>Source Input Picture (encoder)</td><td>8-bit uncompressed data</td></tr><tr><td>0101b</td><td>Reconstructed Scaled Reference Picture</td><td>8-bit data</td></tr></table></td></tr>
<tr><td>2</td><td>31:18</td><td>Height<table><tr><td>Format:</td><td>U14-1 Height</td></tr></table>This field specifies the height of the Picture in units of pixels/residuals. For PLANAR surface formats, this field indicates the height of the Y (luma) plane. Note : Gen7 Video Codecs must program less than and equal to 4K.(In future, it will be ideal to have this field define in a WORD boundary.)AVC - multiple of 2 MB rows for field pictureVC1 - mulitple of 4 pixels for field pictureMPEG2 - multiple of 2 MB rows for field picJPEG - mulitple of integral MCU (8 or 16 pixels) per picture<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>[0,16383]</td><td></td><td>representing heights [1,16384]</td></tr></table></td></tr>
</table>

MFX_SURFACE_STATE

		Programming Notes • For AVC : For frame picture is a multiple of 16; for field picture is a multiple of 32 • For VC1 : For progressive frames, the frame height and frame width is a multiple of 2 pixels. For interlaced frames, the frame height shall be a multiple of 4 pixels, and its width is a multiple of 2 pixels, based on a PLANAR_420 surface. • For SVC : The pixel or residual heights for streamin and streamout.
	17:4	**Width** Format: U14-1 Width This field specifies the width of the Picture in units of pixels/residuals. For PLANAR surface formats, this field indicates the width of the Y (luma) plane. Value: [0,16383] — Name: (none) — Description: representing widths [1,16384] **Programming Notes** • The Width specified by this field multiplied by the pixel size in bytes must be less than or equal to the surface pitch (specified in bytes via the Surface Pitch field). • Width (field value + 1) must be a multiple of 2 for PLANAR_420 • For SVC : the pixel or residual width for streamin and streamout. • MFX HW does not use this field, the picture width is read from IMG State instead, because this field may not equal to the actual picture width. This field is used by the KMD to allocate surface in GTT.
	3:2	**Reserved** Format: MBZ
	1:0	**Cr(V)/Cb(U) Pixel Offset V Direction** Project: All Format: U0.2 exactly as shown in the original spec Specifies the distance to the U/V values with respect to the even numbered Y channels in the V direction **Programming Notes** This field is ignored for all formats except PLANAR_420_8
3	31:28	**Surface Format** Format: U4 Specifies the format of the surface. All of the Y and G channels will use table 0 and all of the Cr/Cb/R/B channels will use table 1.Usage: For 420 planar YUV surface, use 4; for monochrome surfaces, use 12. For monochrome surfaces, hardware ignores control fields for Chroma planes.This field must be set to 4 - PLANAR_420_8, or 12 - Y8_UNORMNot used for MFX, and is ignored. But for JPEG decoding, this field should be programmed to the same format as JPEG_PIC_STATE. For video codec, it should set to 4 always.

MFX_SURFACE_STATE

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

27 **Interleave Chroma**

Format:	Enable

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

Value	Name
1	Enable
0	Disable

26:20 **Reserved**

Format:	MBZ

19:3 **Surface Pitch**

Format:	U17-1 pitch in Bytes

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

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

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

MFX_SURFACE_STATE

DWord	Bit	Description
		For Each SVC SurfaceID: 00b: 8-bit uncompressed pixel or coeff_pred data - pitch >= upper layer pic width aligned to 128-byte tile. 01b: 16-bit uncompressed residual data - pitch >= 2*upper layer pic width aligned to 128-byte tile. 10b: 8-bit uncompressed pixel or coeff_pred data - pitch >= lower layer pic width aligned to 128-byte tile. 11b: 16-bit uncompressed residual data - pitch >= 2*lower layer pic width aligned to 128-byte tile.
	2	**Half Pitch for Chroma** Format: Enable (This field must be set to Disable)This field indicates that the chroma plane(s) will use a pitch equal to half the value specified in the Surface Pitch field. This field is only used for PLANAR surface formats.This field is igored by MFX (unless we support YV12)
	1	**Tiled Surface** Format: Boolean (This field must be set to TRUE: Tiled)This field specifies whether the surface is tiled.This field is ignored by MFX Value 0 – Name: False – Description: Linear Value 1 – Name: True – Description: Tiled **Programming Notes** Linear surfaces can be mapped to Main Memory (uncached) or System Memory (cacheable, snooped). Tiled surfaces can only be mapped to Main Memory.The corresponding cache(s) must be invalidated before a previously accessed surface is accessed again with an altered state of this bit.
	0	**Tile Walk** Format: 3D_Tilewalk (This field must be set to 1: TILEWALK_YMAJOR)This field specifies the type of memory tiling (XMajor or YMajor) employed to tile this surface. See Memory Interface Functions for details on memory tiling and restrictions.This field is ignored when the surface is linear.This field is ignored by MFX. Internally H/W is always treated this set to 1 for all video codec and for JPEG. Value 0h – Name: XMAJOR – Description: TILEWALK_XMAJOR Value 1h – Name: YMAJOR – Description: TILEWALK_YMAJOR **Programming Notes** The corresponding cache(s) must be invalidated before a previously accessed surface is accessed again with an altered state of this bit
4	31	**Reserved** Format: MBZ

MFX_SURFACE_STATE

DWord	Bit	Description
	30:16	**X Offset for U(Cb)** Project: All Format: U15 Pixel Offset This field specifies the horizontal offset in pixels from the Surface Base Address to the start (origin) of the U(Cb) plane or the interleaved UV plane if Interleave Chroma is enabled. This field is only used for PLANAR surface formats. This field must be set to zero.X Offset for U(Cb) in pixel (This field must be zero for NV12 and IMC 1 and 3) **Programming Notes** For PLANAR_420 and PLANAR_422 surface formats, this field must be zero.
	15	**Reserved**
	14:0	**Y Offset for U(Cb)** Project: All Format: U15 Pixel Row Offset This field specifies the veritical offset in rows from the Surface Base Address to the start (origin) of the U(Cb) plane or the interleaved UV plane if Interleave Chroma is enabled. This field is only used for PLANAR surface formats. **Programming Notes** For PLANAR_420 and PLANAR_422 surface formats, this field must be multiple of 16 pixels - i.e. multiple MBs. For JPEG, this field must be a multiple of 16 pixels.
5	31:29	**Reserved**
	28:16	**X Offset for V(Cr)** Format: U13 Offset in Pixels This field must be zero for NV12 and IMC 1 and 3 This field specifies the horizontal offset in pixels from the Surface Base Address to the start (origin) of the V(Cr) plane. This field is only used for PLANAR surface formats with Interleave Chroma disabled. **Programming Notes** For PLANAR_420 and PLANAR_422 surface formats, this field must indicate an even number of pixels.
	15:0	**Y Offset for V(Cr)** Format: U16 Row Offset in Pixels This field specifies the veritical offset in rows from the Surface Base Address to the start (origin) of the V(Cr) plane. This field is only used for PLANAR surface formats with Interleave Chroma disabled. This field is ignored by all video codec, only used by JPEG. **Programming Notes** For PLANAR_420 surface formats, this field must be multiple of 16 pixels - i.e. multiple MBs. For JPEG, this field must be a multiple of 16 pixels.

MFX_VC1_DIRECTMODE_STATE

MFX_VC1_DIRECTMODE_STATE

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2
Exists If:	//VC1 decoding in VLD modes

This is a picture level command and should be issued only once, even for a multi-slices picture. There is only one DMV buffer for read (when processing a B-picture) and one for write (when processing a P-Picture). Each DMV record is 64 bits per MB, to store the top and bottom field MVs (32-bit MVx,y each).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_VC1_DIRECTMODE_STATE Format: OpCode
	26:24	**Media Command Opcode** Default Value: 2h VC1_COMMON Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 2h Format: OpCode
	15:12	**Reserved**
	11:0	**DWord Length** Default Value: 0005h Excludes DWord (0,1) Project: All Format: =n Total Length - 2
1	31:6	**Direct MV Write Buffer Base Address for the Current Picture** This field provides the base address of the DMV write buffer to store the motion vectors decoded in the current picture. It is a private buffer used by the MPR hardware only. Its content is not accessed by software.This buffer must be 64-byte cacheline aligned.The write buffer size is 557,056 bytes for 1 frame. Scalable with frame height, but do not scale with frame width as the hardware assumes frame width (in MBs) fixed at 128 (smallest power of 2 value larger than 120 - 1920x1088 screen resolution).This field is only valid for a P picture

MFX_VC1_DIRECTMODE_STATE

DWord	Bit	Description
	5:0	**Reserved** Project: CHV, BSW Format: MBZ
2 **Project:** CHV, BSW	31:16	**Reserved** Project: CHV, BSW Format: MBZ
	15:0	**Direct MV Write Buffer Base Address for the Current Picture [47:32]** Project: CHV, BSW This field is for the upper range of Direct MV Write Buffer Base Address for the Current Picture. This field is used for 48-bit addressing.
3 **Project:** CHV, BSW	31:15	**Reserved** Project: CHV, BSW Format: MBZ
	14:13	**Reserved**
	12:11	**Reserved**
	10:9	**Reserved**
	8:7	**Direct MV Write Buffer Base Address for the Current Picture - Arbitration Priority Control** Project: CHV, BSW 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
	6:5	**Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for Direct MV Write Buffer for the Current Picture** Project: CHV, BSW This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream. Value / Name: 00b: Use Cacheability Controls from page table 01b: Uncacheable (UC) - non-cacheable 10b: Writethrough (WT) 11b: Writeback (WB)

<table>
<tr><th colspan="3">MFX_VC1_DIRECTMODE_STATE</th></tr>
<tr><td rowspan="3"></td><td>4:3</td><td>Target Cache (TC) for Direct MV Write Buffer for the Current Picture
<table><tr><td>Project:</td><td>CHV, BSW</td></tr></table>
This field allows the choice of LLC vs eLLC for caching
<table><tr><th>Value</th><th>Name</th></tr>
<tr><td>00b</td><td>eLLC Only - not snooped in GT</td></tr>
<tr><td>01b</td><td>LLC Only</td></tr>
<tr><td>10b</td><td>LLC/eLLC Allowed</td></tr>
<tr><td>11b</td><td>L3, LLC, eLLC Allowed</td></tr></table></td></tr>
<tr><td>2</td><td>Encrypted Data for Direct MV Write Buffer for the Current Picture
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>Enable</td></tr></table>
This field controls whether data is decrypted while being read. This field is ignored for writes.</td></tr>
<tr><td>1:0</td><td>Age for QUADLRU (AGE) for Direct MV Write Buffer for the Current Picture
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>Enable</td></tr></table>
This field allows the selection of AGE parameter for a given surface in LLC or eLLC.
<table><tr><th>Value</th><th>Name</th></tr>
<tr><td>11b</td><td>Good chance of generating hits.</td></tr>
<tr><td>10b</td><td>Next good chance of generating hits</td></tr>
<tr><td>01b</td><td>Decent chance of generating hits</td></tr>
<tr><td>00b</td><td>Poor chance of generating hits</td></tr></table></td></tr>
<tr><td rowspan="2">4</td><td>31:6</td><td>Direct MV Read Buffer Base Address for the Reference Picture
This field provides the base address of the DMV buffer for reference picture. It is a private buffer used by the MPR hardware only. Its content is not accessed by software.All these buffers must be 64-byte cacheline aligned.This field is only valid for a B picture.</td></tr>
<tr><td>5:0</td><td>Reserved
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td rowspan="2">5
Project:
CHV,
BSW</td><td>31:16</td><td>Reserved
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table>
Reserved for 64-bit address extension.</td></tr>
<tr><td>15:0</td><td>Direct MV Read Buffer Base Address for the Current Picture [47:32]
<table><tr><td>Project:</td><td>CHV, BSW</td></tr></table>
This field is for the upper range of Direct MV Read Buffer Base Address for the Current Picture. This field is used for 48-bit addressing.</td></tr>
</table>

MFX_VC1_DIRECTMODE_STATE

<table>
<tr><td rowspan="7">6
Project:
CHV, BSW</td><td>31:15</td><td>**Reserved**
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>14:13</td><td>**Reserved**
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>12:11</td><td>**Reserved**
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>10:9</td><td>**Reserved**
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>8:7</td><td>**Direct MV Read Buffer Base Address for the Current Picture - Arbitration Priority Control**
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>U2</td></tr></table>This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>Highest priority</td></tr><tr><td>01b</td><td>Second highest priority</td></tr><tr><td>10b</td><td>Third highest priority</td></tr><tr><td>11b</td><td>Lowest priority</td></tr></table></td></tr>
<tr><td>6:5</td><td>**Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for Direct MV Read Buffer for the Current Picture**
<table><tr><td>Project:</td><td>CHV, BSW</td></tr></table>This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream.<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>Use Cacheability Controls from page table</td></tr><tr><td>01b</td><td>Uncacheable (UC) - non-cacheable</td></tr><tr><td>10b</td><td>Writethrough (WT)</td></tr><tr><td>11b</td><td>Writeback (WB)</td></tr></table></td></tr>
<tr><td>4:3</td><td>**Target Cache (TC) for Direct MV Read Buffer for the Current Picture**
<table><tr><td>Project:</td><td>CHV, BSW</td></tr></table>This field controls the L3$, LLC and eLLC (eDRAM) cacheability for a given surface. Setting of "00" points to PTE settings which defaults to eDRAM (when present). If no eDRAM, the access will be allocated to LLC. Setting of "01", allocates into LLC and victimizes the line to eDRAM. Setting of "10" allows the line to be allocated in either LLC or eDRAM. Setting of "11" is the</td></tr>
</table>

MFX_VC1_DIRECTMODE_STATE

only option for a memory access to be allocated in L3$ as well as LLC/eLLC 00b: eLLC Only ("00" setting points TC selection to PTE which defaults to eLLC) 01b: LLC Only (Works at the allocation time, later victimization from LLC downgrades the line to eLLC if present). 10b: LLC/eLLC Allowed. 11b: L3, LLC, eLLC Allowed.

Value	Name
00b	eLLC Only - not snooped in GT
01b	LLC Only
10b	LLC/eLLC Allowed
11b	L3, LLC, eLLC Allowed

2 **Encrypted Data for Direct MV Read Buffer for the Current Picture**

Project:	CHV, BSW
Format:	Enable

This field controls whether data is decrypted while being read. This field is ignored for writes.

1:0 **Age for QUADLRU (AGE) for Direct MV Read Buffer for the Current Picture**

Project:	CHV, BSW
Format:	Enable

This field allows the selection of AGE parameter for a given surface in LLC or eLLC. If a particular allocation is done at youngest age (“0,1,2”) it tends to stay longer in the cache. This option is given to GFX software to be able to decide which surfaces are more likely to generate HITs, hence need to be replaced least often in caches.

Value	Name
11b	Good chance of generating hits.
10b	Next good chance of generating hits
01b	Decent chance of generating hits
00b	Poor chance of generating hits.

MFX_VC1_PRED_PIPE_STATE

MFX_VC1_PRED_PIPE_STATE

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

This command is used to set the operating states of the MFD Engine beyond the BSD unit. It is used with both VC1 Long and Short format.Driver is responsible to take the intensity compensation enable signal, the LumScale and the LumShift provided from the DXVA2 VC1 interface, and maintain a history of these values for reference pictures. Together with these three parameters specified for the current picture being decoded, driver will derive and supply the above sets of LumScaleX, LumShiftX and intensity compensation enable (single or double, forward or backward) signals. H/W is responsible to take these state values, and use them to build the lookup table (including the derivation of iScale and iShift) for remapping the reference frame pixels, as well as perfoming the actual pixel remapping calculations/process.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_VC1_PRED_PIPE_STATE Format: OpCode
	26:24	**Media Command Opcode** Default Value: 2h VC1_COMMON Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 1h Format: OpCode
	15:12	**Reserved** Project: All Format: MBZ
	11:0	**DWord Length** Default Value: 0004h Excludes DWord (0,1) Project: All Format: =n Total Length - 2
1	31:16	**Reserved**

MFX_VC1_PRED_PIPE_STATE

	15:14	**vin_intensitycomp_Double_FWDen** Format: U2 for forward reference picture only, to enable top or/and bottom of the reference field enable for single compensation. For frame, may only need one bit. This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	13:12	**vin_intensitycomp_Double_BWDen** Format: U2 for backward reference picture only, no double for backward reference. This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	11:10	**vin_intensitycomp_Single_FWDen** Format: U2 for forward reference picture only, to enable top or/and bottom of the reference field enable for single compensation. For frame, may only need one bit. This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	9:8	**vin_intensitycomp_Single_BWDen** Format: U2 for backward reference picture only, no double for backward reference. This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	7:4	**Reference Frame Boundary Replication Mode** Format: U4 This is a bit field with each bit indicating the corresponding picture's boundary replication mode. Bit 11: reference 3 Bit 10: reference 2 Bit 9: reference 1 Bit 8: reference 0 0 = progressive frame replication 1 = interlace frame replication This field is maintained and provided by driver for both long and short VC1 interface format.
	3:0	**Reserved** Format: MBZ
2	31:30	**Reserved** Format: MBZ

MFX_VC1_PRED_PIPE_STATE

	29:24	**LumShift2- single - FWD** Format: U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	23:22	**Reserved** Format: MBZ
	21:16	**LumShift1 - single - FWD** Format: U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	15:14	**Reserved** Format: MBZ
	13:8	**LumScale2 - single - FWD** Format: U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	7:6	**Reserved** Format: MBZ
	5:0	**LumScale1 - Single - FWD** Format: U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
3	31:30	**Reserved** Format: MBZ
	29:24	**LumShift2- double - FWD** Format: U6 This field is maintained andprovided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.

MFX_VC1_PRED_PIPE_STATE		
	23:22	**Reserved** Format: MBZ
	21:16	**LumShift1 - double -FWD** Format: U6 This field is maintained andprovided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	15:14	**Reserved**
	13:8	**LumScale2 - double - FWD** Format: U6 This field is maintained andprovided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	7:6	**Reserved**
	5:0	**LumScale1 - double - FWD** Format: U6 This field is maintained andprovided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
4	31:30	**Reserved**
	29:24	**LumShift2- single - BWD** Format: U6 This field is maintained andprovided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	23:22	**Reserved** Format: MBZ
	21:16	**LumShift1 - single - BWD** Format: U6 This field is maintained andprovided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	15:14	**Reserved**

MFX_VC1_PRED_PIPE_STATE		
	13:8	**LumScale2 - single - BWD** Format: U6 This field is maintained andprovided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	7:6	**Reserved** Format: MBZ
	5:0	**LumScale1 - Single - BWD** Format: U6 This field is maintained andprovided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
5	31:30	**Reserved** Format: MBZ
	29:24	**LumShift2- double - BWD** Format: U6 This field is maintained andprovided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	23:22	**Reserved** Format: MBZ
	21:16	**LumShift1 - double -BWD** Format: U6 This field is maintained andprovided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	15:14	**Reserved** Format: MBZ
	13:8	**LumScale2 - double - BWD** Format: U6 This field is maintained andprovided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.

MFX_VC1_PRED_PIPE_STATE		
	7:6	**Reserved** Format: MBZ
	5:0	**LumScale1 - double - BWD** Format: U6 This field is maintained andprovided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.

MFX_VP8_BSP_BUF_BASE_ADDR_STATE

MFX_VP8_BSP_BUF_BASE_ADDR_STATE

Project: CHV, BSW
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 Video Codec Format: OpCode
	26:24	**Media Command OpCode** Default Value: 4h VP8 Format: OpCode
	23:21	**Sub Opcode A** Default Value: 2h VP8 Common Format: OpCode
	20:16	**Sub Opcode B** Default Value: 3h MFX_VP8_BSP_BUF_BASE_ADDR_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Value | Name | Description 000h | Excludes DWord (0,1) **[Default]** | 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." 008h | | Used for normal encode mode
1	31:6	**Frame Header Base Addr** Format: StreamInAddress[31:6] 64 bytes aligned buffer in linear format. 48-bit Abs. Address StreamIn Surface **Note:**The format is linear vs. tile for better performance.

MFX_VP8_BSP_BUF_BASE_ADDR_STATE		
	5:0	**Reserved** Format: MBZ
2	31:16	**Reserved** Format: MBZ
	15:0	**Frame Header Base Addr - Upper Range**
3	31:9	**Reserved** Format: MBZ
	8:7	**Frame Header Base Addr - Arbitration Priority Control** 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
	6:5	**for FrameHeaderBaseAddr - LLC/eLLC Cacheability Control (LeLLCCC)** This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream. Value / Name / Description: 00b / Cacheable / Use Cacheability Controls from page table 01b / UC / Uncacheable - non-cacheable 10b / WT / Writethrough 11b / WB / Writeback
	4:3	**Frame Header Base Addr - Target Cache (TC)** This field allows the choice of LLC vs eLLC for caching Value / Name: 00b: eLLC Only - not snooped in GT 01b: LLC Only 10b: LLC/eLLC Allowed 11b: L3, LLC, eLLC Allowed
	2	**Frame Header Base Addr - Encrypted Data** Format: Enable This field controls whether data is decrypted while being read. This field is ignored for writes.
	1:0	**Frame Header Base Addr - Age for QUADLRU (AGE)** This field allows the selection of AGE parameter for a given surface in LLC or eLLC. . If a particular allocation is done at youngest age ("3") it tends to stay longer in the cache as compared to older age allocations ("2", "1", or "0"). This option is given to driver to be able to decide which surfaces are more likely to generate HITs, hence need to be replaced least often in caches.

MFX_VP8_BSP_BUF_BASE_ADDR_STATE

Value	Name
00b	Good chance of generating hits
01b	Next good chance of generating hits
10b	Decent chance of generating hits
11b	Poor chance of generating hits

DWord	Bit	Description
4	31:6	**Intermediate Buffer Base Addr**
	5:0	**Reserved**
5	31:16	**Reserved**
	15:0	**Intermediate Buffer Base Addr - Upper Range**
6	31:9	**Reserved**
	8:7	**Intermediate Buffer Base Addr - Arbitration Priority Control**
	6:5	**Intermediate Buffer Base Addr - LLC/eLLC Cacheability Control (LeLLCCC)**
	4:3	**Intermediate Buffer Base Addr- Target Cache (TC)**

DWord 4, 31:6 Intermediate Buffer Base Addr

Format:	StreamInAddress[31:6] 64 bytes aligned buffer in linear format

48-bit AbsAddr StreamIn Surface

Note: The format is linear vs. tile for better performance.

DWord 4, 5:0 Reserved

Format:	MBZ

DWord 5, 31:16 Reserved

Format:	MBZ

DWord 5, 15:0 Intermediate Buffer Base Addr - Upper Range

DWord 6, 31:9 Reserved

Format:	MBZ

DWord 6, 8:7 Intermediate Buffer Base Addr - Arbitration Priority Control

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

DWord 6, 6:5 Intermediate Buffer Base Addr - LLC/eLLC Cacheability Control (LeLLCCC)

This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream.

Value	Name	Description
00b	Cacheable	Use Cacheability Controls from page table
01b	UC	Uncacheable - non-cacheable
10b	WT	Writethrough
11b	WB	Writeback

DWord 6, 4:3 Intermediate Buffer Base Addr- Target Cache (TC)

This field allows the choice of LLC vs. eLLC for caching

Value	Name
00b	eLLC Only - not snooped in GT

MFX_VP8_BSP_BUF_BASE_ADDR_STATE

		01b — LLC Only 10b — LLC/eLLC Allowed 11b — L3, LLC, eLLC Allowed
	2	**Intermediate Buffer Base Addr- Encrypted Data** This field controls whether data is decrypted while being read. This field is ignored for writes.
	1:0	**Intermediate Buffer Base Addr- Age for QUADLRU (AGE)** This field allows the selection of AGE parameter for a given surface in LLC or eLLC. . If a particular allocation is done at youngest age ("3") it tends to stay longer in the cache as compared to older age allocations ("2", "1", or "0"). This option is given to driver to be able to decide which surfaces are more likely to generate HITs, hence need to be replaced least often in caches. **Value** / **Name** 00b — Good chance of generating hits 01b — Next good chance of generating hits 10b — Decent chance of generating hits 11b — Poor chance of generating hits
7	31:0	**Intermediate Buffer Partition-1 Offset** Format: U32 **Programming Notes** All **Intermediate Buffer Partition-[i] Offset** (i = 1 to 8) and **Intermediate Buffer Max Size** need to be cacheline aligned (64Byte aligned).
8	31:0	**Intermediate Buffer Partition-2 Offset** Format: U32
9	31:0	**Intermediate Buffer Partition-3 Offset** Format: U32
10	31:0	**Intermediate Buffer Partition-4 Offset** Format: U32
11	31:0	**Intermediate Buffer Partition-5 Offset** Format: U32
12	31:0	**Intermediate Buffer Partition-6 Offset** Format: U32
13	31:0	**Intermediate Buffer Partition-7 Offset** Format: U32
14	31:0	**Intermediate Buffer Partition-8 Offset** Format: U32
15	31:0	**Intermediate Buffer Max Size** Format: U32

MFX_VP8_BSP_BUF_BASE_ADDR_STATE		
16	31:6	**Final Frame Base Addr** Format: StreamInAddress[31:6] 64 bytes aligned buffer in linear format 48-bit AbsAddr StreamIn Surface **Note:**The format is linear vs. tile for better performance.
	5:0	**Reserved** Format: MBZ
17	31:16	**Reserved** Format: MBZ
	15:0	**Final Frame Base Addr - Upper Range**
18	31:9	**Reserved** Format: MBZ
	8:7	**Final Frame Base Addr - Arbitration Priority Control**
	6:5	**Final Frame Base Addr - LLC/eLLC Cacheability Control (LeLLCCC)** This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream.
	4:3	**Final Frame Base Addr - Target Cache (TC)** This field allows the choice of LLC vs eLLC for caching
	2	**Final Frame Base Addr - Encrypted Data** Format: Enable This field controls whether data is encrypted while being read. This field is ignored for writes.

Final Frame Base Addr - Arbitration Priority Control (bits 8:7):

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

Final Frame Base Addr - LLC/eLLC Cacheability Control (bits 6:5):

Value	Name	Description
00b	Cacheable	Use Cacheability Controls from page table
01b	UC	Uncacheable - non-cacheable
10b	WT	Writethrough
11b	WB	Writeback

Final Frame Base Addr - Target Cache (bits 4:3):

Value	Name
00b	eLLC Only - not snooped in GT
01b	LLC Only
10b	LLC/eLLC Allowed
11b	L3, LLC, eLLC Allowed

MFX_VP8_BSP_BUF_BASE_ADDR_STATE

	1:0	**Final Frame Base Addr - Age for QUADLRU (AGE)** This field allows the selection of AGE parameter for a given surface in LLC or eLLC. . If a particular allocation is done at youngest age ("3") it tends to stay longer in the cache as compared to older age allocations ("2", "1", or "0"). This option is given to driver to be able to decide which surfaces are more likely to generate HITs, hence need to be replaced least often in caches. **Value** / **Name** 00b: Good chance of generating hits 01b: Next good chance of generating hits 10b: Decent chance of generating hits 11b: Poor chance of generating hits
19	31:6	**Reserved** Format: MBZ
	5:0	**FinalFrameByteOffset** Format: U6 Specify byte offset within a 64-byte cacheline where the bitstream should be inserted at.
20	31:6	**Streamout Base Addr** Format: StreamInAddress[31:6] 64 bytes aligned buffer in linear format 48-bit AbsAddr StreamIn Surface **Note:**The format is linear vs. tile for better performance.
	5:4	**Streamout Base Addr - Arbitration Priority Control** 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
	3	**Streamout Base Addr - Encrypted Data** Format: Enable This field controls whether data is encrypted while being read. This field is ignored for writes
	2	**Streamout Base Addr - Graphics Data Type (GFDT)** Format: U1 This field contains the GFDT bit for this surface when writes occur. GFDT can also be set by the GTT. The effective GFDT is the logical OR of this field with the GFDT from the GTT entry. This field is ignored for reads.

MFX_VP8_BSP_BUF_BASE_ADDR_STATE

	1:0	**Streamout Base Addr - Cacheability Control** Format: U2 EnumeratedType This field controls cacheability in the mid-level cache (MLC) and last-level cache (LLC) **Value / Name / Description** 00b / GTT Entry / Use cacheability control bits from GTT entry 01b / Not LLC or MLC / Data is not cached in LLC or MLC 10b / LLC but not MLC / Data is cached in LLC but not MLC 11b / Both LLC and MLC / Data is cached in both LLC and MLC
21	31:16	**Reserved** Format: MBZ
	15:0	**Streamout Base Addr - Upper Range**
22	31:9	**Reserved**
	8:7	**Streamout Base Addr - Arbitration Priority Control** **Value / Name** 00b / Highest priority 01b / Second highest priority 10b / Third highest priority 11b / Lowest priority
	6:5	**Streamout Base Addr - LLC/eLLC Cacheability Control (LeLLCCC)** This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream. **Value / Name / Description** 00b / Cacheable / Use Cacheability Controls from page table 01b / UC / Uncacheable - non-cacheable 10b / WT / Writethrough 11b / WB / Writeback
	4:3	**Streamout Base Addr - Target Cache (TC)** This field allows the choice of LLC vs eLLC for caching **Value / Name** 00b / eLLC Only - not snooped in GT 01b / LLC Only 10b / LLC/eLLC Allowed 11b / L3, LLC, eLLC Allowed
	2	**Streamout Base Addr - Encrypted Data** Format: Enable This field controls whether data is encrypted while being read. This field is ignored for writes.

MFX_VP8_BSP_BUF_BASE_ADDR_STATE

	1:0	**Streamout Base Addr - Age for QUADLRU (AGE)** This field allows the selection of AGE parameter for a given surface in LLC or eLLC. . If a particular allocation is done at youngest age ("3") it tends to stay longer in the cache as compared to older age allocations ("2", "1", or "0"). This option is given to driver to be able to decide which surfaces are more likely to generate HITs, hence need to be replaced least often in caches. **Value** / **Name** 00b: Good chance of generating hits 01b: Next good chance of generating hits 10b: Decent chance of generating hits 11b: Poor chance of generating hits
23	31:6	**Coeff Probs StreamIn Surface** Format: StreamInAddress[31:6] 64 bytes aligned buffer in linear format 48-bit AbsAddr StreamIn Surface **Note:**The format is linear vs. tile for better performance.
	5:4	**Coeff Probs StreamIn Surface - Arbitration Priority Control** 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
	3	**Coeff Probs StreamIn Surface - Encrypted Data** Format: Enable This field controls whether data is encrypted while being read. This field is ignored for writes
	2	**Coeff Probs StreamIn Surface - Graphics Data Type (GFDT)** Format: U1 This field contains the GFDT bit for this surface when writes occur. GFDT can also be set by the GTT. The effective GFDT is the logical OR of this field with the GFDT from the GTT entry. This field is ignored for reads.
	1:0	**Coeff Probs StreamIn Surface - Cacheability Control** Format: U2 EnumeratedType This field controls cacheability in the mid-level cache (MLC) and last-level cache (LLC) **Value** / **Name** / **Description** 00b / GTT Entry / Use cacheability control bits from GTT entry 01b / Not LLC or MLC / Data is not cached in LLC or MLC 10b / LLC but not MLC / Data is cached in LLC but not MLC 11b / Both LLC and MLC / Data is cached in both LLC and MLC

MFX_VP8_BSP_BUF_BASE_ADDR_STATE

DWord	Bit	Description
24	31:16	**Reserved** Format: MBZ
	15:0	**Coeff Probs StreamIn Surface - Upper Range** This field is for the upper range of Coeff Probs[4][8][3][11]
25	31:9	**Reserved** Format: MBZ
	8:7	**Coeff Probs StreamIn Surface - Arbitration Priority Control** Value / Name: 00b: Highest priority 01b: Second highest priority 10b: Third highest priority 11b: Lowest priority
	6:5	**Coeff Probs StreamIn Surface - LLC/eLLC Cacheability Control (LeLLCCC)** This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream. Value / Name / Description: 00b: Cacheable - Use Cacheability Controls from page table 01b: UC - Uncacheable - non-cacheable 10b: WT - Writethrough 11b: WB - Writeback
	4:3	**Coeff Probs StreamIn Surface - Target Cache (TC)** This field allows the choice of LLC vs eLLC for caching Value / Name: 00b: eLLC Only - not snooped in GT 01b: LLC Only 10b: LLC/eLLC Allowed 11b: L3, LLC, eLLC Allowed
	2	**Coeff Probs StreamIn Surface - Encrypted Data** Format: Enable This field controls whether data is encrypted while being read. This field is ignored for writes.
	1:0	**Coeff Probs StreamIn Surface - Age for QUADLRU (AGE)** This field allows the selection of AGE parameter for a given surface in LLC or eLLC. . If a particular allocation is done at youngest age ("3") it tends to stay longer in the cache as compared to older age allocations ("2", "1", or "0"). This option is given to driver to be able to decide which surfaces are more likely to generate HITs, hence need to be replaced least often in caches.

MFX_VP8_BSP_BUF_BASE_ADDR_STATE

		<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>Good chance of generating hits</td></tr><tr><td>01b</td><td>Next good chance of generating hits</td></tr><tr><td>10b</td><td>Decent chance of generating hits</td></tr><tr><td>11b</td><td>Poor chance of generating hits</td></tr></table>
26	31:6	**Token Statistics Surface** <table><tr><td>Format:</td><td>TokenStatisticsAddress[31:6]</td></tr></table><table><tr><td>48-bit Abs. Address StreamIn Surface</td></tr><tr><td>**Note:**The format is linear vs. tile for better performance.</td></tr></table>
	5:4	**Frame Header Base Addr - Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>Highest priority</td></tr><tr><td>01b</td><td>Second highest priority</td></tr><tr><td>10b</td><td>Third highest priority</td></tr><tr><td>11b</td><td>Lowest priority</td></tr></table>
	3	**Token Statistics Surface- Decrypted Data** <table><tr><td>Exists If:</td><td>//Encrypted Data</td></tr><tr><td>Format:</td><td>Enable</td></tr></table>This field controls whether data is decrypted while being read. This field is ignored for writes.
	2	**Token Statistics Surface - Graphics Data Type (GFDT)** <table><tr><td>Format:</td><td>U1</td></tr></table>This field contains the GFDT bit for this surface when writes occur. GFDT can also be set by the GTT. The effective GFDT is the logical OR of this field with the GFDT from the GTT entry. This field is ignored for reads.
	1:0	**Token Statistics Surface - Cacheability Control** This field controls cacheability in the mid-level cache (MLC) and last-level cache (LLC).<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>00b</td><td>GTT Entry</td><td>Use cacheability control bits from GTT entry</td></tr><tr><td>01b</td><td>Not LLC or MLC</td><td>Data is not cached in LLC or MLC</td></tr><tr><td>10b</td><td>LLC but not MLC</td><td>Data is cached in LLC but not MLC</td></tr><tr><td>11b</td><td>Both LLC and MLC</td><td>Data is cached in both LLC and MLC</td></tr></table>
27	31:16	**Reserved**
	15:0	**Token Statistics Surface** This field is for the upper range of Token Statistics Address.

MFX_VP8_BSP_BUF_BASE_ADDR_STATE

28	31:9	**Reserved** Format: MBZ
	8:7	**Token Statistics Surface - Arbitration Priority Control** 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
	6:5	**Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for CoeffProbs StreamIn Surface** This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream. Value / Name / Description: 00b: Use Cacheability Controls from page table 01b: UC: Uncacheable - non-cacheable 10b: WT: Writethrough 11b: WB: Writeback
	4:3	**Token Statistics Surface - Target Cache (TC)** This field allows the choice of LLC vs eLLC for caching. Value / Name: 00b: eLLC Only - not snooped in GT 01b: LLC Only 10b: LLC/eLLC Allowed 11b: L3, LLC, eLLC Allowed
	2	**Token Statistics Surface** Exists If: //Encrypted Data This field controls whether data is decrypted while being read. This field is ignored for writes.
	1:0	**Token Statistics Surface - Age for QUADLRU (AGE)** This field allows the selection of AGE parameter for a given surface in LLC or eLLC. . If a particular allocation is done at youngest age ("3") it tends to stay longer in the cache as compared to older age allocations ("2", "1", or "0"). This option is given to driver to be able to decide which surfaces are more likely to generate HITs, hence need to be replaced least often in caches. Value / Name: 11b: Good chance of generating hits. 10b: Next good chance of generating hits 01b: Decent chance of generating hits 00b: Poor chance of generating hits

<table>
<tr><th colspan="3">MFX_VP8_BSP_BUF_BASE_ADDR_STATE</th></tr>
<tr><td>29</td><td>31:6</td><td>MPC RowStore Surface Address Low
Format: GraphicsAddress[31:6]
48-bit Abs. Address StreamIn/StreamOut Surface. Note: The format is linear vs. tile for better performance.</td></tr>
<tr><td></td><td>5:4</td><td>MPC RowStore Base Addr - Arbitration Priority Control
This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>00b</td><td>Highest priority</td></tr>
<tr><td>01b</td><td>Second highest priority</td></tr>
<tr><td>10b</td><td>Third highest priority</td></tr>
<tr><td>11b</td><td>Lowest priority</td></tr>
</table></td></tr>
<tr><td></td><td>3</td><td>MPC RowStore Surface - Encrypted Data
Exists If: //Encrypted Data
Format: U1
This field controls whether data is decrypted while being read. This field is ignored for writes.</td></tr>
<tr><td></td><td>2</td><td>MPC RowStore Surface Graphics Data Type (GFDT)
Format: U1
This field contains the GFDT bit for this surface when writes occur. GFDT can also be set by the GTT. The effective GFDT is the logical OR of this field with the GFDT from the GTT entry. This field is ignored for reads.</td></tr>
<tr><td></td><td>1:0</td><td>MPC RowStore Surface - Cacheability Control
This field controls cacheability in the mid-level cache (MLC) and last-level cache (LLC).
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>00b</td><td>GTT Entry</td><td>Use cacheability control bits from GTT entry</td></tr>
<tr><td>01b</td><td>Not LLC or MLC</td><td>Data is not cached in LLC or MLC</td></tr>
<tr><td>10b</td><td>LLC but not MLC</td><td>Data is cached in LLC but not MLC</td></tr>
<tr><td>11b</td><td>Both LLC and MLC</td><td>Data is cached in both LLC and MLC</td></tr>
</table></td></tr>
<tr><td>30</td><td>31:16</td><td>Reserved
Format: MBZ</td></tr>
<tr><td></td><td>15:0</td><td>MPC RowStore Surface Address High
Format: GraphicsAddress[47:32]
This field is for the upper range of Token Statistics Address.</td></tr>
<tr><td>31</td><td>31:9</td><td>Reserved
Format: MBZ</td></tr>
</table>

MFX_VP8_BSP_BUF_BASE_ADDR_STATE

	8:7	**MPC RowStore - Arbitration Priority Control** **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

	6:5	**MPC RowStore - Memory Type: LLC/eLLC Cacheability Control** **This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.**

Value	Name	Description
00b	Use Cacheability Controls from page table	
01b	UC	Uncacheable - non-cacheable
10b	WT	Writethrough
11b	WB	Writeback

	4:3	**MPC RowStore - Target Cache**

Format:	U2

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

Value	Name
00b	eLLC Only
01b	LLC Only
10b	LLC/eLLC Allowed
11b	L3, LLC, eLLC Allowed

	2	**MPC RowStore Surface**

Exists If:	//Encrypted Data
Format:	U1

This field controls whether data is decrypted while being read. This field is ignored for writes.

	1:0	**MPC RowStore Surface - Age for QUADLRU (AGE)** **This field allows the selection of AGE parameter for a given surface in LLC or eLLC. . If a particular allocation is done at youngest age ("3") it tends to stay longer in the cache as compared to older age allocations ("2", "1", or "0"). This option is given to driver to be able to decide which surfaces are more likely to generate HITs, hence need to be replaced least often in caches.**

Value	Name
11b	Good chance of generating hits.
10b	Next good chance of generating hits.
01b	Decent chance of generating hits.
00b	Poor chance of generating hits.

MFX_VP8_Encoder_CFG

MFX_VP8_Encoder_CFG

Project:	CHV, BSW
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 and must be issued before MFX_VP8_PIC_STATE.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Video Codec Format: OpCode
	26:24	**Media Command OpCode** Default Value: 4h VP8 Format: OpCode
	23:21	**Sub Opcode A** Default Value: 2h VP8 Common Format: OpCode
	20:16	**Sub Opcode B** Default Value: 1h MFX_VP8_ENCODER_CFG Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n

Value	Name	Description	Project
000h	Excludes DWord (0,1) **[Default]**	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."	
01Ch		Used for normal encode mode	CHV, BSW

<table>
<tr><th colspan="3">MFX_VP8_Encoder_CFG</th></tr>
<tr><td rowspan="10">1</td><td>31:11</td><td>Reserved
Format: MBZ</td></tr>
<tr><td>10</td><td>VBSPunitPowerClock Gating Disable
Project: CHV, BSW
Format: U1
VBSPunit Power Clock Gating Disable.</td></tr>
<tr><td>9</td><td>Compressed Bitstream Output Disable
Project: CHV, BSW
Format: U1
Disable Compressed Bitstream Output.(Both Final Bitstream and Intermediate bit buffer)</td></tr>
<tr><td>7</td><td>Per Segment Delta Qindex / LoopFilter Disable
Project: +, CHV, BSW
Format: U1
Disable Per Segment Delta Qindex / Loop Filter in Rate Control.</td></tr>
<tr><td>6</td><td>Rate Control Initial Pass
Project: CHV, BSW
Format: U1
<table><tr><th>Value</th><th>Name</th></tr><tr><td>1</td><td>Initial pass</td></tr><tr><td>0</td><td>Subsequence Pass(es)</td></tr></table></td></tr>
<tr><td>5</td><td>Skip Final Bitstream when Over / Under flow
Format: U1
Skip Final Bitstream conditionally on Over/Under flow in rate control and intermediate Bit Buffer Overrun.</td></tr>
<tr><td>4</td><td>Update Segment Feature Data Flag
Exists If: //VP8 Encoder
Format: U1
Enable for Frame Header per Segment Quantizer / LoopFilter Update</td></tr>
<tr><td>3</td><td>Bitstream Statistics Output Enable
Enable Bitstream Statistics Output at Memory Surface in MFX_VBSP_BUF_ADDR_STATE DW[26:28]</td></tr>
<tr><td>2</td><td>Token Statistics Output Enable
Enable Token Statistics Output at Memory Surface in MFX_VBSP_BUF_ADDR_STATE DW[26:28]</td></tr>
<tr><td>1</td><td>Final Bitstream Output Disable
Format: U1
Disable Final Bitstream Output.</td></tr>
</table>

MFX_VP8_Encoder_CFG		
	0	**Performance Counter Enable** Format: U1 Enable Performance Counter in Streamout.
2	31:8	**Reserved** Format: MBZ
	7	**Qindex_Clamp_High_mask for overflow** Project: CHV, BSW Format: U1 If current frame is overflow and this mask is set, it would mask out MFX_VP8_Img_Status register. DW1.bit1. In another word, subsequent passes would be skipped.
	6	**Qindex_Clamp_High_mask for underflow** Project: CHV, BSW Format: U1 If current frame is underflow and this mask is set, it would mask out MFX_VP8_Img_Status register. DW1.bit0. In another word, subsequent passes would be skipped
	5	**Final Bistream Buffer Overrun Enable Mask** Project: CHV, BSW Format: U1 Enable Final Bitstream Buffer Overrun detection feature.
	4	**Intermediate Bit Buffer Overrun Enable Mask** Format: U1 Enable Intermediate Bit Buffer Overrun detection feature.
	3	**Max Intra MB Bit Count Check Enable Mask** Format: U1 Enable Max. Intra MB bit count check in Streamout.
	2	**Max Inter MB Bit Count Check Enable Mask** Format: U1 Enable Max. Inter MB bit count check in Streamout.
	1	**Min Frame Bit Count Rate Control Enable Mask** Format: U1 Enable Min. Frame Rate Control. This is a mask bit controlling if the condition of frame level bit count is less than or equal to FrameBitRateMin.

MFX_VP8_Encoder_CFG

Value	Name	Description
1		If (Total Frame Level Bit Counter) =< (Frame Bit Rate Minimum limit) Set bit[0] and bit[1] of MFX_VP8_IMAGE_STATUS Control Register.
0		Do not update bit[0] of MFX_VP8_IMAGE_STATUS Control Register.

Bit 0 — Max Frame bit count Rate Control Enable Mask

Format:	U1

Enable Max. Frame Rate Control. This is a mask bit controlling if the condition of frame level bit count is greater than or equal to FrameBitRateMax.

Value	Name	Description
1		If (Total Frame Level Bit Counter) >= (Frame Bit Rate Maximum Limit) Set bit[0] and bit[1] of MFX_VP8_IMAGE_STATUS control register.
0		Do not update bit[0] of MFX_VP8_IMAGE_STATUS control register.

DWord 3

31:28 — Reserved

27:16 — Max Intra MB Bit Count Limit

Format:	U12

12-bit bit count for Max Intra MB Limit.

15:12 — Reserved

Format:	MBZ

11:0 — Max Inter MB bit count

Format:	U12

12-bit bit count for Max Inter MB Limit.

DWord 4

31 — Frame Bitrate Min Unit Mode

Format:	U1

This field is the Frame Bitrate Minimum Limit Units.

Value	Name	Description
0h	Compatibility Mode	Frame BitRate Min Unit is in old mode **(128b/16Kb)**
1h	New Mode	Frame BitRate Min Unit is in new mode **(32byte/4Kb)**

30 — Frame Bit Rate Min Unit

Format:	U1

This field is Frame Bitrate Minimum Mode.

Value	Name
0	32-B
1	4-KB

29:16 — Frame Bit Rate Min

Format:	U14

If either BRC Underflow or overflow is enabled. Frame Bit Rate Min and Frame Bit Rate Max need to be programmed with unambiguous values

MFX_VP8_Encoder_CFG

DWord	Bit	Description
	15	**Frame Bitrate Max Unit Mode** Format: U1 This field is the Frame Bitrate Maximum Limit Units. **Value** | **Name** | **Description** 0h | Compatibility Mode | Frame BitRate Max Unit is in old mode **(128b/16Kb)** 1h | New Mode | Frame BitRate Max Unit is in new mode **(32byte/4Kb)**
	14	**Frame Bit Rate Max Unit** Format: U1 *This field is Frame Bitrate Maximum Mode* **Value** | **Name** 0 | 32-B 1 | 4-KB
	13:0	**Frame Bit Rate Max** Format: U14 If either BRC Underflow or overflow is enabled. Frame Bit Rate Min and Frame Bit Rate Max need to be programmed with unambiguous values
5	31:24	**Frame Delta QIndex Max[3]** This field is the Frame level delta Qindex for total bit-count above FrameBitRateMax - First 1/8 Region. This field is used to calculate the suggested Frame Qindex into the MFX_VP8_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	**Frame DeltaQ Index Max[2]** This field is the Frame level delta Qindex for bit-count above FrameBitRateMax - Above 1/8 and Below 1/4. This field is used to calculate the suggested Frame Qindex into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is between 1/8 and 1/4 of FrameBitRateMaxDelta above **FrameBitRateMax**; i.e., In the range of ((FrameBitRateMax + FrameBitRateMaxDelta » 3), (FrameBitRateMax+ FrameBitRateMaxDelta » 2)].
	15:8	**Frame Delta QIndex Max[1]** This field is the Frame level delta QINDEX for bit-count above FrameBitRateMax - Above 1/4 and Below 1/2. This field is used to calculate the suggested Frame QINDEX into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is between 1/4 and 1/2 of FrameBitRateMaxDelta above **FrameBitRateMax**; i.e., In the range of [(FrameBitRateMax+ FrameBitRateMaxDelta » 2), (FrameBitRateMax+ FrameBitRateMaxDelta » 1)].

MFX_VP8_Encoder_CFG

	7:0	**Frame Delta QIndex Max [0]** This field is the Frame level delta QINDEX for bit-count above FrameBitRateMax - Above 1/2. This field is used to calculate the suggested Frame QINDEX into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is above FrameBitRateMax by more than half the distance of **FrameBitRateMax**; i.e., In the range of [(FrameBitRateMax + FrameBitRateMaxDelta » 1), ∞ (Infinite)].
6	31:24	**Frame Delta QIndex Min[3]** This field is the Frame level delta QINDEX for total bit-count below FrameBitRateMin - First 1/8 Region. This field is used to calculate the suggested Frame QINDEX into the MFX_VP8_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	**Frame Delta QIndex Min[2]** This field is the Frame level delta QINDEX for bit-count below FrameBitRateMin - Below 1/ 8 and Above 1/4. This field is used to calculate the suggested Frame QINDEX into the MFX_VP8_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	**Frame Delta QIndex Min[1]** This field is the Frame level delta QINDEX for bit-count below FrameBitRateMin- Below 1/4 and Above 1/2. This field is used to calculate the suggested Frame QINDEX into the MFX_VP8_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	**Frame Delta QIndex Min[0]** This field is the Frame Level Delta QINDEX for bit-count below FrameBitRateMin - Below 1/2. This field is used to calculate the suggested Frame QINDEX into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is below FrameBitRateMin by more than half the distance of **FrameBitRateMin**; i.e., In the range of [0, (FrameBitRateMin - FrameBitRateMinDelta » 1).
7	31:0	**Per Segment Frame Delta QIndex Max[1]** Project: CHV, BSW Reserved in CHV, BSW, Need to be programmed with same value as Segment0. Per Segment BRC is not validated in CHV, BSW Fulsim
8	31:0	**Per Segment Frame Delta QIndex Min[1]** Project: CHV, BSW Reserved in CHV, BSW, Need to be programmed with same value as Segment0. Per Segment BRC is not validated in CHV, BSW Fulsim

MFX_VP8_Encoder_CFG		
9	31:0	**Per Segment Frame Delta QIndex Max[2]** Project: CHV, BSW Reserved in CHV, BSW, Need to be programmed with same value as Segment0. Per Segment BRC is not validated in CHV, BSW Fulsim
10	31:0	**Per Segment Frame Delta QIndex Min[2]** Project: CHV, BSW Reserved in CHV, BSW, Need to be programmed with same value as Segment0. Per Segment BRC is not validated in CHV, BSW Fulsim
11	31:0	**Per Segment Frame Delta QIndex Max[3]** Project: CHV, BSW Reserved in CHV, BSW, Need to be programmed with same value as Segment0. Per Segment BRC is not validated in CHV, BSW Fulsim
12	31:0	**Per Segment Frame Delta QIndex Min[3]** Project: CHV, BSW Reserved in CHV, BSW, Need to be programmed with same value as Segment0. Per Segment BRC is not validated in CHV, BSW Fulsim
13	31:24	**Frame Delta Loop Filter Max[3]** Format: U8 This field is the Frame level delta LoopFilter for total bit-count above FrameBitRateMax - First 1/8 region. This field is used to calculate the suggested Frame LoopFilter into the MFX_VP8_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	**Frame Delta Loop Filter Max[2]** Format: U8 This field is the Frame level delta LoopFilter for bit-count above FrameBitRateMax - Above 1/8 and Below 1/4. This field is used to calculate the suggested Frame LoopFilter into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is between 1/8 and 1/4 of FrameBitRateMaxDelta above FrameBitRateMax. i.e., in the range of ((FrameBitRateMax + FrameBitRateMaxDelta » 3) and (FrameBitRateMax + FrameBitRateMaxDelta » 2)].
	15:8	**Frame Delta Loop Filter Max[1]** Format: U8 This field is the Frame level delta LOOPFILTER for bit-count above FrameBitRateMax - Above1/ 4 and Below 1/2.

<table>
<tr><th colspan="3">MFX_VP8_Encoder_CFG</th></tr>
<tr><td></td><td></td><td>This field is used to calculate the suggested Frame LOOPFILTER into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is between 1/4 and1/2 of FrameBitRateMaxDelta above FrameBitRateMax. i.e., in the range of ((FrameBitRateMax+ FrameBitRateMaxDelta » 2) and (FrameBitRateMax+ FrameBitRateMaxDelta » 1)].</td></tr>
<tr><td></td><td>7:0</td><td>Frame Delta Loop Filter Max[0]
Format: U8
This field is the Frame level delta LOOPFILTER for bit-count above FrameBitRateMax - Above 1/2.
This field is used to calculate the suggested Frame LOOPFILTER into the MFX_VP8_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).</td></tr>
<tr><td>14</td><td>31:24</td><td>Frame Delta Loop Filter Min[3]
Format: U8
This field is the Frame level delta LOOPFILTER for total bit-count below FrameBitRateMin - First 1/8 region.
This field is used to calculate the suggested Frame LOOPFILTER into the MFX_VP8_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).</td></tr>
<tr><td></td><td>23:16</td><td>Frame Delta Loop Filter Min[2]
Format: U8
This field is the Frame level delta LOOPFILTER for bit-count below FrameBitRateMin - Below 1/ 8 and Above 1/4.
This field is used to calculate the suggested Frame LOOPFILTER into the MFX_VP8_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)).</td></tr>
<tr><td></td><td>15:8</td><td>Frame Delta Loop Filter Min[1]
Format: U8
This field is the Frame level delta LOOPFILTER for bit-count below FrameBitRateMin- Below 1/4 and Above 1/2.
This field is used to calculate the suggested Frame LOOPFILTER into the MFX_VP8_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) and (FrameBitRateMin - FrameBitRateMinDelta » 2)).</td></tr>
</table>

MFX_VP8_Encoder_CFG

DWord	Bit	Description
	7:0	**Frame Delta Loop Filter Min[0]** Format: U8 This field is the Frame Level Delta LOOPFILTER for bit-count below FrameBitRateMin - Below 1/2. This field is used to calculate the suggested Frame LOOPFILTER into the MFX_VP8_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).
15	31:0	**Per Segment Frame Delta LoopFilter Max[1]** Project: +, CHV, BSW
16	31:0	**Per Segment Frame Delta LoopFilter Min[1]** Project: +, CHV, BSW
17	31:0	**Per Segment Frame Delta LoopFilter Max[2]** Project: +, CHV, BSW
18	31:0	**Per Segment Frame Delta LoopFilter Min[2]** Project: +, CHV, BSW
19	31:0	**Per Segment Frame Delta LoopFilter Max[3]** Project: +, CHV, BSW
20	31:0	**Per Segment Frame Delta LoopFilter Min[3]** Project: +, CHV, BSW
21	31	**Reserved**
	30:16	**FrameBitRateMinDelta** Format: U15 This field is used to select the frame delta QINDEX when FrameBitRateMin Is exceeded. It shares the same FrameBitrateMinUnit. Value: [0-4095] — Name: (none) — Description: When FrameBitrateMinUnit is in Bytes, this range is in Bytes. When FrameBitrateMinUnit is in KB, this range is in KB units.
	15	**Reserved**
	14:0	**Frame Bit Rate Max Delta** Format: U15 This field is used to select the frame delta QINDEX when FrameBitRateMax Is exceeded. It shares the same FrameBitrateMaxUnit. Value: [0-4095] — Name: (none) — Description: When FrameBitrateMinUnit is in Bytes, this range is in Bytes. When FrameBitrateMinUnit is in KB, this range is in KB units.
22	31:24	**Reserved**

MFX_VP8_Encoder_CFG

DWord	Bit	Description
	23	**Show Frame** Format: U1 VP8 Frame Tag, Show Frame Field
	22:20	**Bitstream Format Version** Format: U3 VP8 Frame Tag, Verison Field
	19:18	**Reserved** Format: MBZ
	17:16	**Min Frame WSize Unit** Format: U2 Value / Name / Description: 0h — Compatibility Mode — MinFrameWSizeUnit is in old mode (128b/16Kb) 1h — New Mode — MinFrameWSizeUnit is in new mode (32byte/4Kb)
	15:0	**Min Frame WSize** Exists If: //Encoder Only This field (in Word, 16-bit) is specified to compensate for Intel® Rate Control. Zero padding would be performed.
23	31:16	**Vertical_Size_Code** Format: U16 Frame Tag Vertical Size Code, composed of {VerticalScale[15:14], FrameHeight[13:0]}
	15:0	**Horizontal_Size_Code** Format: U16 Frame Tag Horizontal Size Code, composed of {HorizontalScale[15:14], FrameWidth[13:0]}
24	31:0	**Frame Header Bit Count** Format: U32 Binarized Header Bit Count.
25	31:0	**Frame Header Bin Buffer Qindex Update Pointer** Format: U32 Binarized Header Qindex Update Pointer If Segment Enabled and UpdateSegmentFeature enabled, 4 per segment Qindices would be updated in Binarized header (Only ABS mode supported). Else Base Qindex would be updated
26	31:0	**Frame Header Bin Buffer LoopFilter Update Pointer** Format: U32 Binarized Header LoopFilter Update Pointer If Segment Enabled and UpdateSegmentFeature enabled, 4 per segment LoopFilters would be updated in Binarized header (Only ABS mode supported). Else Base LoopFilter would be updated.

<table>
<tr><th colspan="3">MFX_VP8_Encoder_CFG</th></tr>
<tr><td>27</td><td>31:0</td><td>Frame Header Bin Buffer Token Update Pointer
<table><tr><td>Format:</td><td>U32</td></tr></table>
Binarized Header TokenUpdate Pointer</td></tr>
<tr><td>28</td><td>31:0</td><td>Frame Header Bin Buffer MVUpdate Pointer
<table><tr><td>Format:</td><td>U32</td></tr></table>
Binarized Header MVUpdate Pointer.</td></tr>
<tr><td rowspan="8">29</td><td>31:28</td><td>ClampValues - CV7</td></tr>
<tr><td>27:24</td><td>CV6</td></tr>
<tr><td>23:20</td><td>CV5</td></tr>
<tr><td>19:16</td><td>CV4</td></tr>
<tr><td>15:12</td><td>CV3</td></tr>
<tr><td>11:8</td><td>CV2</td></tr>
<tr><td>7:4</td><td>CV1</td></tr>
<tr><td>3:0</td><td>CV0 - Clamp Value 0
<table><tr><td>Format:</td><td>U4</td></tr></table>
<table>
<tr><td>If the magnitude of coefficients at locations assigned with CV0 (mapping shown below) exceeds 2CV0-1, they are replaced with 2CV0-1. 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).</td></tr>
<tr><td>For 4x4 frame block, each coefficient is mapped to one of the eight CV values as following:
<table>
<tr><td>none</td><td>CV7</td><td>CV5</td><td>CV4</td></tr>
<tr><td>CV7</td><td>CV6</td><td>CV4</td><td>CV3</td></tr>
<tr><td>CV5</td><td>CV4</td><td>CV2</td><td>CV1</td></tr>
<tr><td>CV4</td><td>CV3</td><td>CV1</td><td>CV0</td></tr>
</table></td></tr>
<tr><td>For 4x4 field block, each coefficient is mapped to one of the eight CV values as following:
<table>
<tr><td>none</td><td>CV6</td><td>CV3</td><td>CV1</td></tr>
<tr><td>CV7</td><td>CV6</td><td>CV3</td><td>CV1</td></tr>
<tr><td>CV5</td><td>CV4</td><td>CV2</td><td>CV0</td></tr>
<tr><td>CV5</td><td>CV4</td><td>CV2</td><td>CV0</td></tr>
</table></td></tr>
</table>
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0-15</td><td></td></tr>
</table>
Programming Notes: In this project, the only value permitted for CV7 through CV0 is 0xf The only value permitted for CV7 through CV0 is 0xf</td></tr>
</table>

MFX_VP8_PAK_OBJECT

MFX_VP8_PAK_OBJECT	
Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

The MFX_VP8_PAK_OBJECT command is the second primitive command for the VP8 Encoding Pipeline. The MV Data portion of the bitstream is loaded as indirect data object.Before issuing a MFX_VP8_PAK_OBJECT command, all VP8 MFX states need to be valid; therefore the commands used to set these states need to have been issued prior to the issue of this command. MB record must be consecutive with no gaps, hence we do not need MB(x,y) in each MB command. Internal counter will keep track of the current MB address, starting from the first MB. MFX_VP8_PAK_OBJECT command follows the MbType definition like MFD. Encoding statistical data such as the total size of the output bitstream are provided through MMIO registers. Software may access these registers through MI_STORE_REGISTER_MEM command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_VP8_PAK_OBJECT Format: OpCode
	26:24	**Media Command Opcode** Default Value: 4h VP8_ENC Format: OpCode
	23:21	**SubOpcode A** Default Value: 2h Format: OpCode
	20:16	**SubOpcode B** Default Value: 9h Format: OpCode
	15:12	**Reserved** Project: All Format: MBZ
	11:0	**DWord Length** Default Value: 5h DWORD_COUNT_n Project: All Format: =n Length -2
1	31:30	**Reserved**

MFX_VP8_PAK_OBJECT

	29	**Reserved** Project: CHV, BSW Format: MBZ
	28:10	**Reserved** Project: All Format: MBZ
	9:0	**Indirect PAK-MV Data Length** Format: U10 This field provides the length in bytes of the indirect data, which contains all the MVs for the current MB (in any partitioning and subpartitioning form). A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect PAK-MV Data Start Address field is ignored. This field must have the same alignment as the Indirect PAK-MV Data Start Address. This field must be DW aligned (since each MV is 4 bytes in size).Driver has to derived this field from MVsize (MVquantity in DXVA, exact size) *4 bytes per MV.
2	31:29	**Reserved** Format: MBZ
	28:0	**Indirect PAK-MV Data Start Address Offset** This field specifies the memory starting address (offset) of the MV data to be fetched into PAK Subsystem for processing. This pointer is relative to the MFC Indirect PAK-MV Object Base Address.Hardware ignores this field if indirect data is not present, i.e. the Indirect PAK-MV Data Length is set to 0. It is a Dword aligned address in all AVC encoding configuration, since each MV is 4 bytes in size. Value: [0,512MB) — Name: (none)
3..6	127:0	**Inline Data** All the required MB level controls and parameters for encoding are captured as Inline Data Description - VP8 PAK OBJECT. It has a fixed size of 4 DWs. Its definition is described in the next section.

MFX_VP8_PIC_STATE

MFX_VP8_PIC_STATE		
Project:	CHV, BSW	
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 and must be issued before MFX_VP8_IMG_STATE.		

Programming Notes	Project
Only able to use this instruction for decoder workloads.	CHV, BSW

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Video Codec Format: OpCode
	26:24	**Media Command OpCode** Default Value: 4h VP8 Format: OpCode
	23:21	**Sub Opcode A** Default Value: 0h VP8 Common Format: OpCode
	20:16	**Sub Opcode B** Default Value: 0h MFX_VP8_PIC_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n

Value	Name	Description
000h	Excludes DWord (0,1) **[Default]**	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."
024h		Used for normal decode and encode mode

MFX_VP8_PIC_STATE		
1	31:24	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	23:16	**Frame Height Minus 1** Exists If: //Decoder / Encoder Format: U8 Picture Height in integer number of MBs minus 1, so the min pic height can be program is 16 rows of pixels.
	15:8	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	7:0	**Frame Width Minus 1** Exists If: //Decoder / Encoder Format: U8 Picture Width in integer number of MBs minus 1, so the min pic width can be program is 16 pixels.
2	31:26	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	25:24	**Log2 Num of Partition** Exists If: //Decoder / Encoder Format: U2 Value / Name: 0 – 1 Token partition 1 – 2 Token partition 2 – 4 Token partition 3 – 8 Token partition
	23:19	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	18:16	**Deblock Sharpness Level** Exists If: //Decoder / Encoder Format: U3 Specifiy the sharpness level, as one of the regular deblocking strength control parameters.

MFX_VP8_PIC_STATE

		Programming Notes Set to 0 to disable the use of sharpness control.
	15:14	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	13	**Alternate Ref Pic MV SignBias Flag** Exists If: //Decoder / Encoder Alternate Reference Picture MV sign bias flag, specified for non-key frame only.
	12	**Golden Ref Picture MV SignBias Flag** Exists If: //Decoder / Encoder Golden Reference Picture MV sign bias flag, specified for non-key frame only.
	11	**Mode Reference Loop Filter Delta Enabled** Exists If: //Decoder / Encoder
	10	**MB NoCoeff SkipFlag** Exists If: //Decoder / Encoder Frame level control if Skip MB (with no non-zero coefficient) is allowed or not.
	9	**Update MBSegment Map Flag** Exists If: //Decoder / Encoder
	8	**Segment Enable Flag** Exists If: //Decoder / Encoder

Bit 11 values:

Value	Name	Description
0		Mode or Reference Loop Filter Delta Adjustment for current frame is disabled.
1		Mode or Reference Loop Filter Delta Adjustment for current frame is enabled.

Bit 10 values:

Value	Name	Description
0		All MBs will have its MB level signaling mb_skip_coeff forced to 0. That is, no skip of coefficient record in the bitstream (even their values are all 0s)
1		Skip MB is enabled in the per MB record.

Bit 9 values:

Value	Name	Description
0		Disable segmentation update
1		Enable segmentation update, and to enable reading segment_id for each MB.

Bit 8 values:

Value	Name	Description
0		Disable Segmentation processing in the current frame

MFX_VP8_PIC_STATE

		1 | | Enable Segmentation processing in the current frame
	7	**Segmentation ID StreamIn Enable** Exists If: //Decoder Only Value | Name 0 | StreamIn Disabled 1 | StreamIn Enabled **Programming Notes** When 0, no input needed.
	7:6	**Reserved** Exists If: //Encoder Only Format: MBZ
	6	**Segmentation ID StreamOut Enable** Exists If: //Decoder Only Value | Name 0 | StreamOut Disabled 1 | StreamOut Enabled **Programming Notes** When 0, no output needed.
	5	**sKeyFrameFlag** Exists If: //Decoder / Encoder Value | Name 0 | Non-Key Frame (P-Frame) 1 | Key Frame (I-Frame)
	4	**DBLKFilterType** Exists If: //Decoder / Encoder To specify VP8 Profile of operation. Value | Name | Description 0 | | Use a full feature normal deblocking filter 1 | | Use a simple filter for deblocking
	3:2	**Reserved** Exists If: //Decoder / Encoder

MFX_VP8_PIC_STATE

DWord	Bit	Description
		Format: MBZ
	1	**Chroma Full Pixel MC Filter Mode** Exists If: //Decoder / Encoder To specify VP8 Profile of operation. Value 0: Chroma MC filter operates in sub-pixel mode Value 1: Chroma MC filter only operates in full pixel position, i.e. no sub-pixel interpolation.
	0	**MC Filter Select** Exists If: //Decoder / Encoder To specify VP8 Profile of operation. Value 0: 6-tap filter (regular filter mode) Value 1: 2-tap bilinear filter (simple profile/version mode)
3	31:30	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	29:24	**DBLKFilterLevel for Segment3** Exists If: //Decoder / Encoder Format: U6 Value 0 — Name: Signifies disable in loop deblocking operation — Description: This is used to set a VP8 profile without in loop deblocker. **Programming Notes**: There are max 4 segments per frame, each segment can have its own deblocking filter level. When segmentation is disabled, only segment 0 parameter is used for the entire frame. **Restriction**: For VP8 Encoder: Max (DBLKFilterLevel for Segment0, DBLKFilterLevel for Segment1, DBLKFilterLevel for Segment2, DBLKFilterLevel for Segment3) + Max (RefLFDelta0, RefLFDelta1, RefLFDelta2, RefLFDelta3) + Max (ModeLFDelta0, ModeLFDelta1, ModeLFDelta2, ModeLFDelta3) < 128
	23:22	**Reserved** Exists If: //Decoder / Encoder

MFX_VP8_PIC_STATE

		Format: MBZ
	21:16	**DBLKFilterLevel for Segment2** Exists If: //Decoder / Encoder Format: U6
	15:14	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	13:8	**DBLKFilterLevel for Segment1** Exists If: //Decoder / Encoder Format: U6
	7:6	**Reserved** Exists If: //Decoder / Encoder

DBLKFilterLevel for Segment2:

Value	Name	Description
0	Signifies disable in loop deblocking operation	This is used to set a VP8 profile without in loop deblocker.

Programming Notes
There are max 4 segments per frame, each segment can have its own deblocking filter level. When segmentation is disabled, only segment 0 parameter is used for the entire frame.

Restriction
For VP8 Encoder: Max (DBLKFilterLevel for Segment0, DBLKFilterLevel for Segment1, DBLKFilterLevel for Segment2, DBLKFilterLevel for Segment3) + Max (RefLFDelta0, RefLFDelta1, RefLFDelta2, RefLFDelta3) + Max (ModeLFDelta0, ModeLFDelta1, ModeLFDelta2, ModeLFDelta3) < 128

DBLKFilterLevel for Segment1:

Value	Name	Description
0	Signifies disable in loop deblocking operation	This is used to set a VP8 profile without in loop deblocker.

Programming Notes
There are max 4 segments per frame, each segment can have its own deblocking filter level. When segmentation is disabled, only segment 0 parameter is used for the entire frame.

Restriction
For VP8 Encoder: Max (DBLKFilterLevel for Segment0, DBLKFilterLevel for Segment1, DBLKFilterLevel for Segment2, DBLKFilterLevel for Segment3) + Max (RefLFDelta0, RefLFDelta1, RefLFDelta2, RefLFDelta3) + Max (ModeLFDelta0, ModeLFDelta1, ModeLFDelta2, ModeLFDelta3) < 128

<table>
<tr><th colspan="3">MFX_VP8_PIC_STATE</th></tr>
<tr><td></td><td></td><td><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>5:0</td><td>DBLKFilterLevel for Segment0
<table><tr><td>Exists If:</td><td>//Decoder / Encoder</td></tr><tr><td>Format:</td><td>U6</td></tr></table>
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td>Signifies disable in loop deblocking operation</td><td>This is used to set a VP8 profile without in loop deblocker.</td></tr></table>
<table><tr><th>Programming Notes</th></tr><tr><td>There are max 4 segments per frame, each segment can have its own deblocking filter level. When segmentation is disabled, only segment 0 parameter is used for the entire frame.</td></tr></table>
<table><tr><th>Restriction</th><th>Project</th></tr><tr><td>For VP8 Encoder: Max (DBLKFilterLevel for Segment0, DBLKFilterLevel for Segment1, DBLKFilterLevel for Segment2, DBLKFilterLevel for Segment3) + Max (RefLFDelta0, RefLFDelta1, RefLFDelta2, RefLFDelta3) + Max (ModeLFDelta0, ModeLFDelta1, ModeLFDelta2, ModeLFDelta3) < 128</td><td>CHV, BSW</td></tr></table></td></tr>
<tr><td>4
Project:
CHV, BSW</td><td>31</td><td>Reserved
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Exists If:</td><td>//Encoder Only</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>30:24</td><td>Seg 3 Qindex
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Exists If:</td><td>//Encoder Only</td></tr><tr><td>Format:</td><td>U7</td></tr></table>
Quantizer Value for Segment ID 3</td></tr>
<tr><td></td><td>23</td><td>Reserved
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Exists If:</td><td>//Encoder Only</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>22:16</td><td>Seg 2 Qindex
<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Exists If:</td><td>//Encoder Only</td></tr><tr><td>Format:</td><td>U7</td></tr></table>
Quantizer Value for Segment ID 2</td></tr>
<tr><td></td><td>15</td><td>Reserved
<table><tr><td>Project:</td><td>CHV, BSW</td></tr></table></td></tr>
</table>

MFX_VP8_PIC_STATE

		Exists If: //Encoder Only Format: MBZ
	14:8	**Seg 1 Qindex** Project: CHV, BSW Exists If: //Encoder Only Format: U7 Quantizer Value for Segment ID 1
	7	**Reserved** Project: CHV, BSW Exists If: //Encoder Only Format: MBZ
	31:0	**Reserved** Project: CHV, BSW Exists If: //Decoder Only Format: MBZ
	6:0	**Seg 0 Qindex** Project: CHV, BSW Exists If: //Encoder Only Format: U7 Quantizer Value for Segment ID 0. **Programming Notes** This is the **[Default]** Qindex
5	31:29	**Reserved** Exists If: //Encoder Only Format: MBZ
	28	**UVac Qindex Delta Sign** Project: CHV, BSW Exists If: //Encoder Only Format: U1 Sign of Quantization index delta for UVac
	27:24	**UVac QindexDelta** Project: CHV, BSW Exists If: //Encoder Only Format: U4 Absolute Quantization index delta for UVac
	23:21	**Reserved**

MFX_VP8_PIC_STATE		
		Project: CHV, BSW Exists If: //Encoder Only Format: MBZ
	20	**UVdc Qindex Delta Sign** Project: CHV, BSW Exists If: //Encoder Only Format: U1 Sign of Quantization index delta for UVdc
	19:16	**UVdc Qindex Delta** Project: CHV, BSW Exists If: //Encoder Only Format: U4 Absolute Quantization index delta for UVdc
	15:13	**Reserved** Project: CHV, BSW Exists If: //Encoder Only Format: MBZ
	12	**Y2ac Qindex Sign** Project: CHV, BSW Exists If: //Encoder Only Format: U1 Sign of Quantization index delta for Y2ac
	11:8	**Y2ac Qindex Delta** Project: CHV, BSW Exists If: //Encoder Only Format: U4 Absolute Quantization index delta for Y2ac
	7:5	**Reserved** Project: CHV, BSW Exists If: //Encoder Only Format: MBZ
	4	**Y2ac Qindex Delta Sign**

MFX_VP8_PIC_STATE		
		Project: CHV, BSW Exists If: //Encoder Only Format: U1 Sign of Quantization index delta for Y2dc This is the **[Default]** Qindex Delta Sign
	31:0	**Reserved** Exists If: //Decoder Only Format: MBZ
	3:0	**Y2dc Qindex Delta** Project: CHV, BSW Exists If: //Encoder Only Format: U4 Absolute Quantization index delta for Y2dc This is the **[Default]** Qindex Delta
6	31:5	**Reserved** Exists If: //Encoder Only Format: MBZ
	4	**Y1dc Qindex Delta Sign** Project: CHV, BSW Exists If: //Encoder Only Format: U1 Sign of Quantization index delta for Y1dc This is the **[Default]**Qindex Delta Sign
	31:0	**Reserved** Project: CHV, BSW Exists If: //Decoder Only Format: MBZ
	3:0	**Y1dc Qindex Delta** Project: CHV, BSW Exists If: //Encoder Only Absolute Quantization index delta for Y1dc This is the **[Default]**Qindex Delta
7	31:15	**Reserved**

MFX_VP8_PIC_STATE		
Project: CHV, BSW		Project: CHV, BSW Exists If: //Encoder Only Format: MBZ
	14:8	**Clamp Qindex high** Project: CHV, BSW Exists If: //Encoder Only Format: U7 Maximum Clamp Value for Qindex used in quantization.
	7	**Reserved** Project: CHV, BSW Exists If: //Encoder Only Format: MBZ
	31:0	**Reserved** Project: CHV, BSW Exists If: //Decoder Only Format: MBZ
	6:0	**Clamp Qindex Low** Project: CHV, BSW Exists If: //Encoder Only Format: U7 Minimum Clamp Value for Qindex used in quantization.
8	31:25	**Reserved** Exists If: //Decoder Only Format: MBZ
	24:16	**Quantizer Value [1][BlockType3=UVAC]** Exists If: //Decoder Only Format: U9 **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
	15:9	**Reserved** Exists If: //Decoder Only Format: MBZ
	31:0	**Reserved**

MFX_VP8_PIC_STATE

		Exists If: //Encoder Only Format: MBZ
	8:0	**Quantizer Value [1][BlockType2=UVDC]** Exists If: //Decoder Only **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
9	31:25	**Reserved** Exists If: //Decoder Only Format: MBZ
	24:16	**Quantizer Value [1][BlockType5=Y2AC]** Exists If: //Decoder Only Format: U9 **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
	15:9	**Reserved** Exists If: //Decoder Only Format: MBZ
	31:0	**Reserved** Exists If: //Encoder Only Format: MBZ
	8:0	**Quantizer Value [1][BlockType4=Y2DC]** Exists If: //Decoder Only **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
10	31:25	**Reserved** Exists If: //Decoder Only Format: MBZ
	24:16	**Quantizer Value [2][BlockType1=Y1AC]** Exists If: //Decoder Only Format: U9 **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
	15:9	**Reserved** Exists If: //Decoder Only Format: MBZ
	31:0	**Reserved** Exists If: //Encoder Only Format: MBZ
	8:0	**Quantizer Value [2][BlockType0=Y1DC]**

MFX_VP8_PIC_STATE		
		Exists If: //Decoder Only **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
11	31:25	**Reserved** Exists If: //Decoder Only Format: MBZ
	24:16	**Quantizer Value [2][BlockType3=UVAC]** Exists If: //Decoder Only Format: U9 **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
	15:9	**Reserved** Exists If: //Decoder Only Format: MBZ
	31:0	**Reserved** Exists If: //Encoder Only Format: MBZ
	8:0	**Quantizer Value [2][BlockType2=UVDC]** Exists If: //Decoder Only **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
12	31:25	**Reserved** Exists If: //Decoder Only Format: MBZ
	24:16	**Quantizer Value [2][BlockType5=Y2AC]** Exists If: //Decoder Only Format: U9 **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
	15:9	**Reserved** Exists If: //Decoder Only Format: MBZ
	31:0	**Reserved** Exists If: //Encoder Only Format: MBZ
	8:0	**Quantizer Value [2][BlockType4=Y2DC]** Exists If: //Decoder Only **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
13	31:25	**Reserved**

MFX_VP8_PIC_STATE

		Exists If: //Decoder Only Format: MBZ
	24:16	**Quantizer Value [3][BlockType1=Y1AC]** Exists If: //Decoder Only Format: U9 **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
	15:9	**Reserved** Exists If: //Decoder Only Format: MBZ
	31:0	**Reserved** Exists If: //Encoder Only Format: MBZ
	8:0	**Quantizer Value [3][BlockType0=Y1DC]** Exists If: //Decoder Only **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
14	31:25	**Reserved** Exists If: //Decoder Only Format: MBZ
	24:16	**Quantizer Value [3][BlockType3=UVAC]** Exists If: //Decoder Only Format: U9 **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
	15:9	**Reserved** Exists If: //Decoder Only Format: MBZ
	31:0	**Reserved** Exists If: //Encoder Only Format: MBZ
	8:0	**Quantizer Value [3][BlockType2=UVDC]** Exists If: //Decoder Only **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
15	31:25	**Reserved**

MFX_VP8_PIC_STATE

DWord	Bit	Description
		Exists If: //Decoder Only Format: MBZ
	24:16	**Quantizer Value [3][BlockType5=Y2AC]** Exists If: //Decoder Only Format: U9 **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
	15:9	**Reserved** Exists If: //Decoder Only Format: MBZ
	31:0	**Reserved** Exists If: //Encoder Only Format: MBZ
	8:0	**Quantizer Value [3][BlockType4=Y2DC]** Exists If: //Decoder Only **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
16	31:6	**CoeffProbability StreamIn Base Address** Exists If: //Decoder Only Format: StreamInAddress[31:6] 64 bytes aligned buffer in linear format. (not tile for better performance) It is specified for non-key frame only. It is the final computed probability table for parsing Coeff in the bitstream. The buffer is unsigned 8-bit * 1056 entries (CoeffProbs[4][8][3][11].
	31:0	**Reserved** Exists If: //Encoder Only Format: MBZ
	5:0	**Reserved** Project: CHV, BSW Exists If: //Decoder Only Format: MBZ
17 **Project:** CHV, BSW	31:16	**Reserved** Project: CHV, BSW Exists If: //Decoder Only Format: MBZ
	31:0	**Reserved**

MFX_VP8_PIC_STATE

		Project: CHV, BSW Exists If: //Encoder Only Format: MBZ
	15:0	**CoeffProbability StreamIn Address** Project: CHV, BSW Exists If: //Decoder Only This field is for the upper range of CoeffProbability StreamIn Address
18 **Project:** CHV, BSW	31:15	**Reserved** Project: CHV, BSW Exists If: //Decoder Only Format: MBZ
	14:13	**Reserved** Project: CHV, BSW Exists If: //Decoder Only Format: MBZ
	12:11	**Reserved** Project: CHV, BSW Exists If: //Decoder Only Format: MBZ
	10:9	**Reserved** Project: CHV, BSW Exists If: //Decoder Only Format: MBZ
	8:7	**CoeffProbability StreamIn - Arbitration Priority Control** Project: CHV, BSW Exists If: //Decoder Only 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
	6:5	**Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for CoeffProbability StreamIn**

MFX_VP8_PIC_STATE

		Address
		Project: CHV, BSW Exists If: //Decoder Only This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream.
	4:3	**CoeffProbability StreamIn Address - Target Cache (TC)**
	2	**Reserved**
	31:0	**Reserved**
	1:0	**CoeffProbability StreamIn Address - Age for QUADLRU (AGE)**

Address

Project:	CHV, BSW
Exists If:	//Decoder Only

This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream.

Value	Name	Description	Exists If
00b	Use Cacheability Controls from page table		//Decoder Only
01b	UC	Uncacheable - non-cacheable	
10b	WT	Writethrough	
11b	WB	Writeback	

4:3 CoeffProbability StreamIn Address - Target Cache (TC)

Project:	CHV, BSW
Exists If:	//Decoder Only

This field allows the choice of LLC vs eLLC for caching

Value	Name
00b	eLLC Only - not snooped in GT
01b	LLC Only
10b	LLC/eLLC Allowed
11b	L3, LLC, eLLC Allowed

2 Reserved

Project:	CHV, BSW

31:0 Reserved

Project:	CHV, BSW
Exists If:	//Encoder Only
Format:	MBZ

1:0 CoeffProbability StreamIn Address - Age for QUADLRU (AGE)

Project:	CHV, BSW
Exists If:	//Decoder Only

This field allows the selection of AGE parameter for a given surface in LLC or eLLC.

Value	Name
11b	Good chance of generating hits.
10b	Next good chance of generating hits
01b	Decent chance of generating hits
00b	Poor chance of generating hits

MFX_VP8_PIC_STATE

DWord	Bit	Description
19	31:24	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	23:16	**MBSegmentIDTreeProbs[2]** Exists If: //Decoder / Encoder Format: U8 MBSegmentIDTreeProbs[2:0] probability tree table for CPBAC parsing Segment_ID of each MB.
	15:8	**MBSegmentIDTreeProbs[1]** Exists If: //Decoder / Encoder Format: U8 MBSegmentIDTreeProbs[2:0] probability tree table for CPBAC parsing Segment_ID of each MB.
	7:0	**MBSegmentIDTreeProbs[0]** Exists If: //Decoder / Encoder Format: U8 MBSegmentIDTreeProbs[2:0] probability tree table for CPBAC parsing Segment_ID of each MB.
20	31:24	**MBNoCoeffSkipFalseProb** Exists If: //Decoder / Encoder Format: U8 8-bit probability value for CPBAC parsing of the MBNoCoeffSkip Flag in the bistream.
	23:16	**IntraMBProb** Exists If: //Decoder / Encoder Format: U8 8-bit probability value for CPBAC parsing of the intra or inter MB type flag in the bitstream.
	15:8	**InterPredFromLastRefProb** Exists If: //Decoder / Encoder Format: U8 8-bit probability value for CPBAC parsing of the flag in the bitstream that determines which reference frame to be used for the current MB motion compensation.
	7:0	**InterPredFromGRefRefProb** Exists If: //Decoder / Encoder Format: U8 8-bit probability value for CPBAC parsing of the flag in the bitstream that determines which reference frame to be used for the current MB motion compensation.
21	31:24	**YModeProb[3]** Exists If: //Decoder / Encoder

MFX_VP8_PIC_STATE

DWord	Bit	Description
		Format: U8 YModeProb[3:0] probability tree table for CPBAC parsing Luma MBType of each MB.
	23:16	**YModeProb[2]** Exists If: //Decoder / Encoder Format: U8 YModeProb[3:0] probability tree table for CPBAC parsing Luma MBType of each MB.
	15:8	**YModeProb[1]** Exists If: //Decoder / Encoder Format: U8 YModeProb[3:0] probability tree table for CPBAC parsing Luma MBType of each MB.
	7:0	**YModeProb[0]** Exists If: //Decoder / Encoder Format: U8 YModeProb[3:0] probability tree table for CPBAC parsing Luma MBType of each MB.
22	31:24	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	23:16	**UVModeProb[2]** Exists If: //Decoder / Encoder Format: U8 UVModeProb[2:0] probability tree table for CPBAC parsing Chroma MBType of each MB.
	15:8	**UVModeProb[1]** Exists If: //Decoder / Encoder Format: U8 UVModeProb[2:0] probability tree table for CPBAC parsing Chroma MBType of each MB.
	7:0	**UVModeProb[0]** Exists If: //Decoder / Encoder Format: U8 UVModeProb[2:0] probability tree table for CPBAC parsing Chroma MBType of each MB.
23	31:24	**MVUpdateProbs[0][3]** Exists If: //Decoder / Encoder

MFX_VP8_PIC_STATE

DWord	Bit	Description
		Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	23:16	**MVUpdateProbs[0][2]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	15:8	**MVUpdateProbs[0][1]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	7:0	**MVUpdateProbs[0][0]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
24	31:24	**MVUpdateProbs[0][7]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	23:16	**MVUpdateProbs[0][6]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	15:8	**MVUpdateProbs[0][5]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	7:0	**MVUpdateProbs[0][4]** Exists If: //Decoder / Encoder

MFX_VP8_PIC_STATE		
		Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
25	31:24	**MVUpdateProbs[0][11]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB.
	23:16	**MVUpdateProbs[0][10]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB.
	15:8	**MVUpdateProbs[0][9]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB.
	7:0	**MVUpdateProbs[0][8]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB.
26	31:24	**MVUpdateProbs[0][15]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	23:16	**MVUpdateProbs[0][14]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	15:8	**MVUpdateProbs[0][13]** Exists If: //Decoder / Encoder

MFX_VP8_PIC_STATE		
		Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	7:0	**MVUpdateProbs[0][12]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
27	31:24	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	23:16	**MVUpdateProbs[0][18]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	15:8	**MVUpdateProbs[0][17]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	7:0	**MVUpdateProbs[0][16]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
28	31:24	**MVUpdateProbs[1][3]** Exists If: //Decoder Only Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	23:16	**MVUpdateProbs[1][2]**

MFX_VP8_PIC_STATE		
		Exists If: //Decoder Only Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	15:8	**MVUpdateProbs[1][1]** Exists If: //Decoder Only Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	7:0	**MVUpdateProbs[1][0]** Exists If: //Decoder Only Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
29	31:24	**MVUpdateProbs[1][7]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	23:16	**MVUpdateProbs[1][6]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	15:8	**MVUpdateProbs[1][5]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	7:0	**MVUpdateProbs[1][4]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
30	31:24	**MVUpdateProbs[1][11]**

MFX_VP8_PIC_STATE

		Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	23:16	**MVUpdateProbs[1][10]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	15:8	**MVUpdateProbs[1][9]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	7:0	**MVUpdateProbs[1][8]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
31	31:24	**MVUpdateProbs[1][15]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	23:16	**MVUpdateProbs[1][14]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	15:8	**MVUpdateProbs[1][13]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	7:0	**MVUpdateProbs[1][12]**

MFX_VP8_PIC_STATE		
		Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
32	31:24	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	23:16	**MVUpdateProbs[1][18]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	15:8	**MVUpdateProbs[1][17]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	7:0	**MVUpdateProbs[1][16]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
33	31	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	30:24	**RefLFDelta3 (for ALTREF FRAME)** Exists If: //Decoder / Encoder Format: S6 2's Compliment Delta value for reference frame based adjustment of the MB-level's filter level value. RefLFDeltas [ref_frametype = 0 to 3] **Programming Notes** Please note that although RefDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.

MFX_VP8_PIC_STATE

		Restriction For VP8 Encoder: Max (DBLKFilterLevel for Segment0, DBLKFilterLevel for Segment1, DBLKFilterLevel for Segment2, DBLKFilterLevel for Segment3) + Max (RefLFDelta0, RefLFDelta1, RefLFDelta2, RefLFDelta3) + Max (ModeLFDelta0, ModeLFDelta1, ModeLFDelta2, ModeLFDelta3) < 128
	23	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	22:16	**RefLFDelta2 (for GOLDEN FRAME)** Exists If: //Decoder / Encoder Format: S6 2's Compliment Delta value for reference frame based adjustment of the MB-level's filter level value. RefLFDeltas [ref_frametype = 0 to 3] **Programming Notes** Please note that although RefDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude. **Restriction** / **Project** For VP8 Encoder: Max (DBLKFilterLevel for Segment0, DBLKFilterLevel for Segment1, DBLKFilterLevel for Segment2, DBLKFilterLevel for Segment3) + Max (RefLFDelta0, RefLFDelta1, RefLFDelta2, RefLFDelta3) + Max (ModeLFDelta0, ModeLFDelta1, ModeLFDelta2, ModeLFDelta3) < 128 / CHV, BSW
	15	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	14:8	**RefLFDelta1 (for LAST FRAME)** Exists If: //Decoder / Encoder Format: S6 2's Compliment Delta value for reference frame based adjustment of the MB-level's filter level value. RefLFDeltas [ref_frametype = 0 to 3] **Programming Notes** Please note that although RefDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude. **Restriction** / **Project**

MFX_VP8_PIC_STATE

DWord	Bit	Description
		For VP8 Encoder: Max (DBLKFilterLevel for Segment0, DBLKFilterLevel for Segment1, DBLKFilterLevel for Segment2, DBLKFilterLevel for Segment3) + Max (RefLFDelta0, RefLFDelta1, RefLFDelta2, RefLFDelta3) + Max (ModeLFDelta0, ModeLFDelta1, ModeLFDelta2, ModeLFDelta3) < 128 — CHV, BSW
	7	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	6:0	**RefLFDelta0 (for INTRA FRAME)** Exists If: //Decoder / Encoder Format: S6 2's Compliment Delta value for reference frame based adjustment of the MB-level's filter level value. RefLFDeltas [ref_frametype = 0 to 3] **Programming Notes** Please note that although RefDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude. **Restriction** / **Project** For VP8 Encoder: Max (DBLKFilterLevel for Segment0, DBLKFilterLevel for Segment1, DBLKFilterLevel for Segment2, DBLKFilterLevel for Segment3) + Max (RefLFDelta0, RefLFDelta1, RefLFDelta2, RefLFDelta3) + Max (ModeLFDelta0, ModeLFDelta1, ModeLFDelta2, ModeLFDelta3) < 128 — CHV, BSW
34	31	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	30:24	**ModeLFDelta3 (for SPLITMV mode)** Exists If: //Decoder / Encoder Format: S6 2's Compliment Delta value for mode based adjustment of the MB-level's filter level value. ModeLFDeltas[MB_Type = 0 to 3] **Programming Notes** Please note that although ModeLFDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude. **Restriction** / **Project**

MFX_VP8_PIC_STATE

		For VP8 Encoder: Max (DBLKFilterLevel for Segment0, DBLKFilterLevel for Segment1, DBLKFilterLevel for Segment2, DBLKFilterLevel for Segment3) + Max (RefLFDelta0, RefLFDelta1, RefLFDelta2, RefLFDelta3) + Max (ModeLFDelta0, ModeLFDelta1, ModeLFDelta2, ModeLFDelta3) < 128	CHV, BSW
	23	**Reserved** Exists If: //Decoder / Encoder Format: MBZ	
	22:16	**ModeLFDelta2 (for Nearest, Near and New mode)** Exists If: //Decoder / Encoder Format: S6 2's Compliment Delta value for mode based adjustment of the MB-level's filter level value. ModeLFDeltas[MB_Type = 0 to 3] **Programming Notes** Please note that although ModeLFDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude. **Restriction** / **Project** For VP8 Encoder: Max (DBLKFilterLevel for Segment0, DBLKFilterLevel for Segment1, DBLKFilterLevel for Segment2, DBLKFilterLevel for Segment3) + Max (RefLFDelta0, RefLFDelta1, RefLFDelta2, RefLFDelta3) + Max (ModeLFDelta0, ModeLFDelta1, ModeLFDelta2, ModeLFDelta3) < 128	CHV, BSW
	15	**Reserved** Exists If: //Decoder / Encoder Format: MBZ	
	14:8	**ModeLFDelta1(for ZEROMV mode)** Exists If: //Decoder / Encoder Format: S6 2's Compliment Delta value for mode based adjustment of the MB-level's filter level value. ModeLFDeltas[MB_Type = 0 to 3] **Programming Notes** Please note that although ModeLFDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude. **Restriction**	**Project**

MFX_VP8_PIC_STATE

		For VP8 Encoder: Max (DBLKFilterLevel for Segment0, DBLKFilterLevel for Segment1, DBLKFilterLevel for Segment2, DBLKFilterLevel for Segment3) + Max (RefLFDelta0, RefLFDelta1, RefLFDelta2, RefLFDelta3) + Max (ModeLFDelta0, ModeLFDelta1, ModeLFDelta2, ModeLFDelta3) < 128 — CHV, BSW
	7	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	6:0	**ModeLFDelta0 (for B_PRED mode)** Exists If: //Decoder / Encoder Format: S6 2's Compliment Delta value for mode based adjustment of the MB-level's filter level value. ModeLFDeltas[MB_Type = 0 to 3] **Programming Notes** Please note that although ModeLFDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude. **Restriction** / **Project** For VP8 Encoder: Max (DBLKFilterLevel for Segment0, DBLKFilterLevel for Segment1, DBLKFilterLevel for Segment2, DBLKFilterLevel for Segment3) + Max (RefLFDelta0, RefLFDelta1, RefLFDelta2, RefLFDelta3) + Max (ModeLFDelta0, ModeLFDelta1, ModeLFDelta2, ModeLFDelta3) < 128 — CHV, BSW
35 **Project:** CHV, BSW	31:0	**Segmentation ID Stream Base Address** Project: CHV, BSW Exists If: //Decoder Only Format: StreamAddress[31:0] 64 bytes linear aligned buffer It is specified when **SegmentationIDStreamInEnable** or **SegmentationIDStreamOutEnable** is specified. **Programming Notes** Each cache has only 8 bits for 4 segmentation ID from 4 continuous MBs.
36 **Project:** CHV, BSW	31:16	**Reserved** Project: CHV, BSW Exists If: //Decoder Only Format: MBZ
	15:0	**Segmentation ID Stream Base Address [47:32]**

MFX_VP8_PIC_STATE

		Project: CHV, BSW Exists If: //Decoder Only This field is for the upper range of **Segmentation ID Stream Base Address**
37 **Project:** CHV, BSW	31:15	**Reserved** Project: CHV, BSW Exists If: //Decoder Only Format: MBZ
	14:13	**Reserved** Project: CHV, BSW Exists If: //Decoder Only Format: MBZ
	12:11	**Reserved** Project: CHV, BSW Exists If: //Decoder Only Format: MBZ
	10:9	**Reserved** Project: CHV, BSW Format: MBZ
	8:7	**Segmentation ID Stream - Arbitration Priority Control** Project: CHV, BSW Exists If: //Decoder Only 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
	6:5	**Memory Type: LLC/eLLC Cacheability Control (LeLLCCC) for Segmentation ID Stream Base Address** Project: CHV, BSW Exists If: //Decoder Only This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LeLLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream. Value / Name / Description

MFX_VP8_PIC_STATE

00b	Use Cacheability Controls from page table	
01b	UC	Uncacheable - non-cacheable
10b	WT	Writethrough
11b	WB	Writeback

4:3

Target Cache (TC) Segmentation ID Stream Base Address

Project:	CHV, BSW
Exists If:	//Decoder Only

This field controls the L3$, LLC and eLLC (eDRAM) cacheability for a given surface. Setting of "00" points to PTE settings which defaults to eDRAM (when present). If no eDRAM, the access will be allocated to LLC. Setting of "01", allocates into LLC and victimizes the line to eDRAM. Setting of "10" allows the line to be allocated in either LLC or eDRAM. Setting of "11" is the only option for a memory access to be allocated in L3$ as well as LLC/eLLC

00b: eLLC Only *("00" setting points TC selection to PTE which defaults to eLLC)*

01b: LLC Only (*Works at the allocation time, later victimization from LLC downgrades the line to eLLC if present*).

10b: LLC/eLLC Allowed.

11b: L3, LLC, eLLC Allowed.

Errata CHV:A-E (FIXED BY:G0 Stepping):

For all system that does NOT use SVM (i.e. coherent L3$ surfaces), back snoops from LLC has to be disabled (**Dis_GtCvUpdtOnRd = “1”**). Than target Cache settings can be programmed as POR requirements of L3/LLC/eDRAM caching.

For all systems that does use SVM (i.e. coherent L3$ surfaces), the recomended setting would be "00" in target cache settings. In case of L3 surfaces, the performance has to be tuned between "00" and "11" setting based on the benefits of L3 caching outweighting the degradation of backsnoops.

Post G0-stepping, the above w/a for coherent L3$ surfaces is not needed.

Value	Name
00b	eLLC Only - not snooped in GT
01b	LLC Only
10b	LLC/eLLC Allowed
11b	L3, LLC, eLLC Allowed

2

Encrypted Data Segmentation ID Stream Base Address

Project:	CHV, BSW
Exists If:	//Decoder Only
Format:	Enable

This field controls whether data is decrypted while being read. This field is ignored for writes.

MFX_VP8_PIC_STATE

	1:0	**Age for QUADLRU (AGE) Segmentation ID Stream Base Address**

Project:	CHV, BSW
Exists If:	//Decoder Only

This field allows the selection of AGE parameter for a given surface in LLC or eLLC. If a particular allocation is done at youngest age ("0,1,2") it tends to stay longer in the cache. This option is given to GFX software to be able to decide which surfaces are more likely to generate HITs, hence need to be replaced least often in caches.

Value	Name
11b	Good chance of generating hits.
10b	Next good chance of generating hits
01b	Decent chance of generating hits
00b	Poor chance of generating hits

MFX_WAIT

MFX_WAIT		
Project:	CHV, BSW	
Source:	VideoCS	
Length Bias:	1	

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

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

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

MI_ARB_CHECK

MI_ARB_CHECK

Project:	CHV, BSW
Source:	VideoEnhancementCS
Length Bias:	1

Description
The MI_ARB_CHECK is used to check for a change in arbitration. If executed as part of a Ring Buffer the command checks the UHPTR valid bit and if set the head of the ring will jump to the value of the head pointer programmed in the UHPTR.

Programming Notes
This instruction cannot be placed in a batch buffer.
If execlist is enabled, there is a pending execution list and this command is parsed, then the command streamer will preempt the current context and start executing the new execution list.

DWord	Bit	Description
0	31:29	**MI Instruction Type** Default Value: 0h MI_INSTRUCTION Format: OpCode
	28:23	**MI Instruction Opcode** Default Value: 05h MI_ARB_CHECK Format: OpCode
	22:0	**Reserved** Project: All Format: MBZ

MI_ARB_CHECK

MI_ARB_CHECK

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	1

Description	Project
The MI_ARB_CHECK is used to check for a change in arbitration. If executed as part of a Ring Buffer the command checks the UHPTR valid bit and if set the head of the ring will jump to the value of the head pointer programmed in the UHPTR.	

Programming Notes	Project
This instruction cannot be placed in a batch buffer.	
If execlist is enabled, there is a pending execution list and this command is parsed, then the command streamer will preempt the current context and start executing the new execution list.	CHV, BSW

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_INSTRUCTION Format: OpCode
	28:23	**MI Command Opcode** Default Value: 05h MI_ARB_CHECK Format: OpCode
	22:0	**Reserved** Format: MBZ

MI_ARB_CHECK

MI_ARB_CHECK

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	1

Description

The MI_ARB_CHECK instruction is used to check the ring buffer double buffered head pointer (register UHPTR). This instruction can be used to pre-empt the current execution of the ring buffer. Note that the valid bit in the updated head pointer register needs to be set for the command streamer to be pre-empted.

Programming Notes

Ring Buffer mode of scheduling:

- The current head pointer is loaded with the updated head pointer register independent of the location of the updated head.
- If the current head pointer and the updated head pointer register are equal, hardware will automatically reset the valid bit corresponding to the UHPTR.
- For pre-emption, the wrap count in the ring buffer head register is no longer maintained by hardware. The hardware updates the wrap count to the value in the UHPTR register.

Execlist mode of scheduling:
MI_ARB_CHK will be used to indicate a command boundary on which Preemption will be honored by Command Streamer in the execlist mode of operation. UHPTR is ignored when processing MI_ARB_CHK in execlist mode.

This instruction can be in either a ring buffer or batch buffer.

MI_ARB_CHK command must not be programmed in INDIRECT_CTX and BB_PER_CTX_PTR buffers.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 05h MI_ARB_CHECK Format: OpCode
	22:0	**Reserved** Format: MBZ

MI_ARB_CHECK

MI_ARB_CHECK

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	1

Description
The MI_ARB_CHECK is used to check for a change in arbitration. If executed as part of a Ring Buffer the command checks the UHPTR valid bit and if set the head of the ring will jump to the value of the head pointer programmed in the UHPTR.

Programming Notes
This instruction cannot be placed in a batch buffer.
If execlist is enabled, there is a pending execution list and this command is parsed, then the command streamer will preempt the current context and start executing the new execution list.

DWord	Bit	Description
0	31:29	**MI Instruction Type** Default Value: 0h MI_INSTRUCTION Format: OpCode
	28:23	**MI Instruction Opcode** Default Value: 05h MI_ARB_CHECK Format: OpCode
	22:0	**Reserved** Format: MBZ

MI_ARB_ON_OFF

MI_ARB_ON_OFF

Project:	CHV, BSW
Source:	CommandStreamer
Length Bias:	1

The MI_ARB_ON_OFF instruction is used to disable/enable context switching. This instruction can be used to prevent submission of a new execlist from interrupting a command sequence, however lite restore preemption is allowed with in the arbitration disabled command execution zone. Note that context switching will remain disabled until re-enabled through use of this command. This command will also prevent a switch in the case of waiting on events, running out of commands. These will effectively hang the device if allowed to occur while arbitration is off (context switching is disabled.) This command should always be used as an off-on pair with the sequence of instructions to be protected from context switch between MI_ARB_OFF and MI_ARB_ON. Software must use this arbitration control with caution since it has the potential to increase the response time of the Render Engine to pre-emption requests. This is a privileged command; it will not be effective (will be converted to a no-op) if executed from within a non-privileged batch buffer.

Execution List Mode of Scheduling: The MI_ARB_ON_OFF instruction is used to disable/enable context switching. Context swiching could be either due to preemption or un-succesfull wait for events or semaphore waits. This instruction can be used to prevent submission of a new execlist from interrupting a command sequence, however lite restore preemption is allowed with in the arbitration disabled command execution zone. Note that context switching will remain disabled until re-enabled through use of this command. This command will also prevent a switch in the case of waiting on events, running out of commands. These will effectively hang the device if allowed to occur while arbitration is off (context switching is disabled.)
Ring Buffer Mode of Scheduling: The MI_ARB_ON_OFF instruction is used to disable preemption on the preemptable commands. SW can explicitly make section of commands in a command buffer non-preemptable by sandwitching them between ARB_OFF and ARB_ON, HW will ingore preemption request (UHPTR Valid) until arbitration is enabled.

Programming Notes
This command must be always be programmed in pairs of off/on in the same command dispatch. Sequence of instructions to be protected from cntext switch or preemption must be programmed between the MI_ARB_OFF and MI_ARB_ON. Software must use this arbitration control with caution since it has the potential to increase the response time of the Render Engine to pre-emption requests. This is a privileged command; it will not be effective (will be converted to a no-op) if executed from within a non-privileged batch buffer.
HW doesn't treat Arbitration Disabled as equivalent to "Inhibit Synchronous Context Switch" set in CTXT_SR_CTL register. Power management optimizations (RDOP on WT4EVT) available on setting "Inhibit Synchronous Context Switch" are not enabled by default on Arbitration Disabled. SW must explicitly program "Inhibit Synchronous Switch" when Arbitration Disabled to enable power management optimizations.

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

MI_ARB_ON_OFF		
	28:23	**MI Command Opcode** Default Value: 08h MI_ARB_ON_OFF Format: OpCode
	22:1	**Reserved** Format: MBZ
	0	**Arbitration Enable** Format: Enable This field enables or disables context switches due to pre-emption (a new execlist).

MI_ATOMIC

MI_ATOMIC

Project:	CHV, BSW
Source:	PRM
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 Vol1i L3 URB [CHV, BSW] B-spec 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 QWORD or DWORD 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

- 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 header must be programmed accordingly.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td>Command Type
<table>
<tr><td>Default Value:</td><td>0h MI_COMMAND</td></tr>
<tr><td>Format:</td><td>OpCode</td></tr>
</table></td></tr>
<tr><td></td><td>28:23</td><td>MI Command Opcode
<table>
<tr><td>Default Value:</td><td>2Fh MI_ATOMIC</td></tr>
<tr><td>Format:</td><td>OpCode</td></tr>
</table></td></tr>
<tr><td></td><td>22</td><td>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.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Per Process Graphics Address</td><td></td></tr>
<tr><td>1h</td><td>Global Graphics Address</td><td>This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.</td></tr>
</table></td></tr>
<tr><td></td><td>21</td><td>Reserved
<table>
<tr><td>Source:</td><td>BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS</td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>21</td><td>Post-Sync Operation
<table>
<tr><td>Source:</td><td>RenderCS</td></tr>
</table>
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>No Post Sync Operation</td><td>Command is executed as usual.</td></tr>
<tr><td>1h</td><td>Post Sync Operation</td><td>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 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
</td></tr>
</table></td></tr>
</table>

MI_ATOMIC

			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.
This bit must not be set due to known HW issues. OR "Post Sync Operation" feature in MI_ATOMIC can be enabled provided "CS Stall" bit is always set in all PIPE_CONTROL commands programmed with "Post Sync Operation" set to "Report Time Stamp" or "Report PS Depth Count".

Workaround
Workaround: "Post Sync Operation" bit must not be set when MI_ATOMIC command is programmed by GPGPU and MEDIA workloads (i.e when PIPELINE_SELECT command is set to GPGPU or MEDIA). This is to WA FFDOP CG issue, this WA need not be implemented when FF_DOP_CG is disabled via "Fixed Function DOP Clock Gate Disable" bit in RC_PSMI_CTRL register.

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.
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

MI_ATOMIC

		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** Project: CHV, BSW Source: RenderCS, BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS **Description** When "Data Size" is QWORD or DWORD 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 Vol1i L3 URB [CHV, BSW] B-spec for atomic opcode corresponding to an atomic operation. **Programming Notes** Atomic Opcode must not be set to 0x00 (no-atomic).
	7:0	**DWord Length** Format: =n Total Length - 2. Excludes DWord (0,1). Value / Name / Exists If: 1h / **[Default]** / ([Inline Data]==0) 9h / / ([Inline Data]==1)
1	31:2	**Memory Address**

MI_ATOMIC

		Project: All 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. **Programming Notes** — Memory Address must be qword aligned for all dword atomic operations. Upper Dword of the memory location should be initialized to 0x0. — **Project:** CHV, BSW
	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.

MI_ATOMIC		
9	31:0	**Operand1 Data Dword 3** Format: U32 Dword3 of Operand1 when Inline Data mode is set.
10	31:0	**Operand2 Data Dword 3** Format: U32 Dword3 of Operand2 when Inline Data mode is set.

MI_BATCH_BUFFER_END

MI_BATCH_BUFFER_END		
Project:	CHV, BSW	
Source:	VideoEnhancementCS	
Length Bias:	1	
The MI_BATCH_BUFFER_END command is used to terminate the execution of commands stored in a batch buffer initiated using a MI_BATCH_BUFFER_START command.		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 0Ah MI_BATCH+_BUFFER_END Format: OpCode
	22:0	**Reserved** Project: All Format: MBZ

MI_BATCH_BUFFER_END

MI_BATCH_BUFFER_END		
Project: CHV, BSW Source: BlitterCS Length Bias: 1		
The MI_BATCH_BUFFER_END command is used to terminate the execution of commands stored in a batch buffer initiated using a MI_BATCH_BUFFER_START command.		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 0h MI_COMMAND
	28:23	**MI Command Opcode** Default Value: 0Ah MI_ BATCH_BUFFER_END
	22:0	**Reserved** Project: All Format: MBZ

MI_BATCH_BUFFER_END

MI_BATCH_BUFFER_END

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	1

The MI_BATCH_BUFFER_END command is used to terminate the execution of commands stored in a batch buffer initiated using a MI_BATCH_BUFFER_START command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 0Ah MI_ BATCH_BUFFER_END Format: OpCode
	22:0	**Reserved** Format: MBZ

MI_BATCH_BUFFER_END

<table>
<tr><th colspan="3">MI_BATCH_BUFFER_END</th></tr>
<tr><td colspan="3">Project: CHV, BSW
Source: VideoCS
Length Bias: 1</td></tr>
<tr><td colspan="3">The MI_BATCH_BUFFER_END command is used to terminate the execution of commands stored in a batch buffer initiated using a MI_BATCH_BUFFER_START command.</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="3">0</td><td>31:29</td><td>Command Type
Default Value: 0h MI_COMMAND
Format: OpCode</td></tr>
<tr><td>28:23</td><td>MI Command Opcode
Default Value: 0Ah MI_BATCH+_BUFFER_END
Format: OpCode</td></tr>
<tr><td>22:0</td><td>Reserved
Format: MBZ</td></tr>
</table>

MI_BATCH_BUFFER_START

MI_BATCH_BUFFER_START

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The MI_BATCH_BUFFER_START command is used to initiate the execution of commands stored in a *batch* buffer. For restrictions on the location of batch buffers, see Batch Buffers in the Device Programming Interface chapter of *MI Functions*. The batch buffer can be specified as privileged or non-privileged, determining the operations considered valid when initiated from within the buffer and any attached (chained) batch buffers. See Batch Buffer Protection in the Device Programming Interface chapter of *MI Functions*.

Programming Notes

- A batch buffer initiated with this command must end either with a MI_BATCH_BUFFER_END command or by chaining to another batch buffer with an MI_BATCH_BUFFER_START command.
- It is essential that the address location beyond the current page be populated inside the GTT. HW performs over-fetch of the command addresses and any over-fetch requires a valid TLB entry. A single extra page beyond the batch buffer is sufficient.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 31h MI_BATCH_BUFFER_START Format: OpCode
	22	**2nd Level Batch Buffer** The command streamer contains three storage elements; one for the ring head address, one for the batch head address, and one for the 2nd level batch head address. When performing batch buffer chaining, hardware simply updates the head pointer of the 1st level batch address storage. There is no stack in hardware. When this bit is set, hardware uses the 2nd level batch head address storage element. Upon MI_BATCH_BUFFER_END, it will automatically return to the 1st (traditional) level batch buffer address. this allows hardware to mimic a simple 3-level stack.

Value	Name	Description
0h	1st level batch	Place the batch buffer address in the 1st (traditional) level batch address storage element.
1h	2nd level batch	Place the batch buffer address in the 2nd-level batch address storage element.

MI_BATCH_BUFFER_START

	Programming Notes Within a second level batch buffer there can't be any chained batch buffers. MI_BATCH_BUFFER_START command is not allowed inside a second level batch buffer.
21:17	**Reserved** Format: MBZ
16	**Add Offset Enable** Format: Enable If this bit is set then the value stored in the BB_OFFSET MMIO register will be added to the Batch Buffer Start Address and the summation will be used as the address to fetch from memory. **Specific to the render command stream only.**
15	**Predication Enable** This bit is used to enable predication of this command. If this bit is set and Bit 0 of the Predicate Result-1 register is clear, this command is ignored. Otherwise the command is performed normally. **Specific to the Render command stream only.**
14:12	**Reserved**
11	**Reserved**
10	**Resource Streamer Enable** Format: Enable When this bit is set, the Resource Streamer will execute the batch buffer. When this bit is clear the Resource Streamer will not execute the batch buffer. **Specific to the Render command stream only.**
9	**Reserved**
8	**Address Space Indicator** Batch buffers accessed via PPGTT are considered as non-privileged. Certain operations (e.g., MI_STORE_DATA_IMM commands to GGTT memory) are prohibited within non-privileged buffers. More details mentioned in User Mode Privileged command section. When MI_BATCH_BUFFER_START command is executed from within a batch buffer (i.e., is a "chained" or "second level" batch buffer command), the current active batch buffer's "Address Space Indicator" and this field determine the "Address Space Indicator" of the next buffer in the chain. • Chained or Second level batch buffer can be in GGTT or PPGTT if the parent batch buffer is in GGTT. • Chained or Second level batch buffer can only be in PPGTT if the parent batch buffer is in PPGTT. This is enforced by Hardware. **Value** | **Name** | **Description** 0h | GGTT | This batch buffer is located in GGTT memory and is privileged. 1h | PPGTT | This batch buffer is located in PPGTT memory and is Non-Privileged. **Programming Notes** This field must be '0' unless the Per-Process GTT Enable is '1'

MI_BATCH_BUFFER_START		
	7:0	**DWord Length** Default Value: 1h Format: =n Total - Bias. Excludes DWord (0,1).
1	31:2	**Batch Buffer Start Address** Format: GraphicsAddress[31:2]BatchBuffer This field specifies Bits 31:2 of the starting address of the batch buffer.
	1:0	**Reserved** Format: MBZ
2	31:16	**Reserved**
	15:0	**Batch Buffer Start Address High** Format: GraphicsAddress[47:32]BatchBuffer This field specifies the 4GB aligned base address of Gfx 4GB virtual address spece within the hosts 64-bit virtual address space.

MI_BATCH_BUFFER_START

MI_BATCH_BUFFER_START		
Project:	CHV, BSW	
Source:	VideoEnhancementCS	
Length Bias:	2	

The MI_BATCH_BUFFER_START command is used to initiate the execution of commands stored in a *batch buffer*. For restrictions on the location of batch buffers, see Batch Buffers in the Device Programming Interface chapter of *MI Functions*.
The batch buffer can be specified as secure or non-secure, determining the operations considered valid when initiated from within the buffer and any attached (chained) batch buffers. See Batch Buffer Protection in the Device Programming Interface chapter of *MI Functions*.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 31h MI_BATCH_BUFFER_START Format: OpCode
	22	**2nd Level Batch Buffer** The command streamer contains 3 storage elements; 1 for the ring head address, 1 for the batch head address, and 1 for the 2nd level batch head address. When performing batch buffer chaining, hardware simply updates the head pointer of the 1st level batch address storage. There is no stack in hardware. When this bit is set, hardware uses the 2nd level batch head address storage element. Upon MI_BATCH_BUFFER_END, it will automatically return to the 1st (traditional) level batch buffer address. this allows hardware to mimic a simple 3 level stack.
	21:13	**Reserved** Format: MBZ
	12	**Reserved**
	11:9	**Reserved** Format: MBZ
	8	**Address Space Indicator** Project: CHV, BSW Batch buffers accessed via PPGTT are considered as non-privileged. Certain operations (e.g.,

Bit 22 values:

Value	Name	Description
0h	1st level batch	Place the batch buffer address in the 1st (traditional) level batch address storage element
1h	2nd level batch	Place the batch buffer address in the 2nd level batch address storage element

MI_BATCH_BUFFER_START

<table>
<tr><td></td><td></td><td>
MI_STORE_DATA_IMM commands to GGTT memory) are prohibited within non-privileged buffers. More details mentioned in User Mode Privileged command section. When MI_BATCH_BUFFER_START command is executed from within a batch buffer (i.e., is a "chained" or "second level" batch buffer command), the current active batch buffer's "Address Space Indicator" and this field determine the "Address Space Indicator" of the next buffer in the chain.

Chained or Second level batch buffer can be in GGTT or PPGTT if the parent batch buffer is in GGTT.
Chained or Second level batch buffer can only be in PPGTT if the parent batch buffer is in PPGTT. This is enforced by Hardware.

<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>GGTT</td><td>This batch buffer is located in GGTT memory and is privileged.</td></tr>
<tr><td>1h</td><td>PPGTT</td><td>This batch buffer is located in PPGTT memory and is Non-Privileged.</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>This field must be '0' unless the Per-Process GTT Enable is '1'</td></tr>
</table>
</td></tr>
<tr><td></td><td>7:0</td><td>
DWord Length (Excludes D-Word 0,1) = 0
<table>
<tr><th>Value</th><th>Name</th><th>Project</th></tr>
<tr><td>1h</td><td>Excludes DWord (0,1) [Default]</td><td>CHV, BSW</td></tr>
</table>
</td></tr>
<tr><td>1</td><td>31:2</td><td>
Batch Buffer Start Address
<table>
<tr><td>Format:</td><td>GraphicsAddress[31:2]</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>
A batch buffer initiated with this command must end either with a MI_BATCH_BUFFER_END command or by chaining to another batch buffer with an MI_BATCH_BUFFER_START command.
The selection of PPGTT vs. GGTT for the batch buffer is determined by the Buffer Security Indicator (bit 8).
</td></tr>
</table>
</td></tr>
<tr><td></td><td>1:0</td><td>
Reserved
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table>
</td></tr>
<tr><td>2
Project:
CHV, BSW</td><td>31:16</td><td>
Reserved
<table>
<tr><td>Project:</td><td>CHV, BSW</td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
</table>
</td></tr>
<tr><td></td><td>15:0</td><td>
Batch Buffer Start Address High
<table>
<tr><td>Project:</td><td>CHV, BSW</td></tr>
<tr><td>Format:</td><td>GraphicsAddress[47:32]BatchBuffer</td></tr>
</table>
This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space.
</td></tr>
</table>

MI_BATCH_BUFFER_START

MI_BATCH_BUFFER_START

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

The MI_BATCH_BUFFER_START command is used to initiate the execution of commands stored in a batch buffer. For restrictions on the location of batch buffers, see Batch Buffers in the Device Programming Interface chapter of MI Functions.The batch buffer can be specified as secure or non-secure, determining the operations considered valid when initiated from within the buffer and any attached (chained) batch buffers. See Batch Buffer Protection in the Device Programming Interface chapter of MI Functions.

Programming Notes

- Batch buffers referenced with physical addresses must not extend beyond the end of the starting physical page (can't span physical pages). However, a batch buffer initiated using a physical address can chain to another buffer in another physical page.
- A batch buffer initiated with this command must end either with a MI_BATCH_BUFFER_END command or by chaining to another batch buffer with an MI_BATCH_BUFFER_START command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 31h MI_BATCH_BUFFER_START Format: OpCode
	22	**2nd Level Batch Buffer** Project: CHV, BSW The command streamer contains 3 storage elements; 1 for the ring head address, 1 for the batch head address, and 1 for the 2nd level batch head address. When performing batch buffer chaining, hardware simply updates the head pointer of the 1st level batch address storage. There is no stack in hardware. When this bit is set, hardware uses the 2nd level batch head address storage element. Upon MI_BATCH_BUFFER_END, it will automatically return to the 1st (traditional) level batch buffer address. this allows hardware to mimic a simple 3 level stack. Value | Name | Description: 0h | 1st level batch | Place the batch buffer address in the 1st (traditional) level batch address storage element 1h | 2nd level batch | Place the batch buffer address in the 2nd level batch address storage element
	21:9	**Reserved**

MI_BATCH_BUFFER_START

Bit 8

Address Space Indicator

Project:	CHV, BSW

Batch buffers accessed via PPGTT are considered as non-privileged. Certain operations (e.g., MI_STORE_DATA_IMM commands to GGTT memory) are prohibited within non-privileged buffers. More details mentioned in User Mode Privileged command section. When MI_BATCH_BUFFER_START command is executed from within a batch buffer (i.e., is a "chained" or "second level" batch buffer command), the current active batch buffer's "Address Space Indicator" and this field determine the "Address Space Indicator" of the next buffer in the chain.

- Chained or Second level batch buffer can be in GGTT or PPGTT if the parent batch buffer is in GGTT.
- Chained or Second level batch buffer can only be in PPGTT if the parent batch buffer is in PPGTT. This is enforced by Hardware.

Value	Name	Description
0h	GGTT	This batch buffer is located in GGTT memory and is privileged.
1h	PPGTT	This batch buffer is located in PPGTT memory and is Non-Privileged.

Programming Notes
This field must be '0' unless the Per-Process GTT Enable is '1'

Bit 7:0

DWord Length

Format:	=n

Total - Bias

Value	Name	Project
1h	Excludes DWord (0,1) **[Default]**	CHV, BSW

DWord 1, Bit 31:2

Batch Buffer Start Address

Format:	GraphicsAddress[31:2]BatchBuffer

This field specifies Bits 31:2 of the starting address of the batch buffer.

DWord 1, Bit 1:0

Reserved

Format:	MBZ

DWord 2 (Project: CHV, BSW), Bit 31:16

Reserved

Project:	CHV, BSW
Format:	MBZ

DWord 2 (Project: CHV, BSW), Bit 15:0

Batch Buffer Start Address High

Project:	CHV, BSW
Format:	GraphicsAddress[47:32]BatchBuffer

This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space.

MI_BATCH_BUFFER_START

MI_BATCH_BUFFER_START

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

The MI_BATCH_BUFFER_START command is used to initiate the execution of commands stored in a batch buffer. For restrictions on the location of batch buffers, see Batch Buffers in the Device Programming Interface chapter of MI Functions.The batch buffer can be specified as secure or non-secure, determining the operations considered valid when initiated from within the buffer and any attached (chained) batch buffers. See Batch Buffer Protection in the Device Programming Interface chapter of MI Functions.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 31h MI_BATCH_BUFFER_START Format: OpCode
	22	**2nd Level Batch Buffer** Project: CHV, BSW The command streamer contains 3 storage elements; 1 for the ring head address, 1 for the batch head address, and 1 for the 2nd level batch head address. When performing batch buffer chaining, hardware simply updates the head pointer of the 1st level batch address storage. There is no stack in hardware. When this bit is set, hardware uses the 2nd level batch head address storage element. Upon MI_BATCH_BUFFER_END, it will automatically return to the 1st (traditional) level batch buffer address. this allows hardware to mimic a simple 3 level stack. Value / Name / Description: 0h / 1st level batch / Place the batch buffer address in the 1st (traditional) level batch address storage element 1h / 2nd level batch / Place the batch buffer address in the 2nd level batch address storage element **Programming Notes** • 2nd level batch buffer chaining is not supported.
	21:10	**Reserved**
	9	**Reserved**

MI_BATCH_BUFFER_START

DWord	Bit	Description
	8	**Address Space Indicator**

Batch buffers accessed via PPGTT are considered as non-privileged. Certain operations (e.g., MI_STORE_DATA_IMM commands to GGTT memory) are prohibited within non-privileged buffers. More details mentioned in User Mode Privileged command section. When MI_BATCH_BUFFER_START command is executed from within a batch buffer (i.e., is a "chained" or "second level" batch buffer command), the current active batch buffer's "Address Space Indicator" and this field determine the "Address Space Indicator" of the next buffer in the chain.

- Chained or Second level batch buffer can be in GGTT or PPGTT if the parent batch buffer is in GGTT.
- Chained or Second level batch buffer can only be in PPGTT if the parent batch buffer is in PPGTT. This is enforced by Hardware.

Value	Name	Description
0h	GGTT	This batch buffer is located in GGTT memory and is privileged.
1h	PPGTT	This batch buffer is located in PPGTT memory and is Non-Privileged.

Programming Notes
This field must be '0' unless the Per-Process GTT Enable is '1'.

DWord	Bit	Description
	7:0	**DWord Length**

Format:	=n Total Length - 2

Value	Name	Project
1h	Excludes DWord (0,1) **[Default]**	CHV, BSW

DWord	Bit	Description
1	31:2	**Batch Buffer Start Address**

Format:	GraphicsAddress[31:2]

Programming Notes
• A batch buffer initiated with this command must end either with a MI_BATCH_BUFFER_END command or by chaining to another batch buffer with an MI_BATCH_BUFFER_START command. • The selection of PPGTT vs. GGTT for the batch buffer is determined by the Buffer Security Indicator (bit8).

DWord	Bit	Description
1	1:0	**Reserved**
2 **Project:** CHV, BSW	31:16	**Reserved**
2 **Project:** CHV, BSW	15:0	**Batch Buffer Start Address High**

Project:	CHV, BSW
Format:	GraphicsAddress[47:32]BatchBuffer

This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space.

MI_CLFLUSH

MI_CLFLUSH

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

Flushes out the page given in the command out to system memory. This command is specific to the render engine and is not privileged.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 27h Store DW MI_CLFLUSH Format: OpCode
	22	**Use Global GTT**
	21:10	**Reserved** Format: MBZ
	9:0	**DWord Length** Format: n Total Length - 2
1	31:12	**Page Base Address** Format: GraphicsAddress[31:12] 4KB aligned Page Address which software requires hardware to flush to DRAM.
	11:6	**Starting Cacheline Offset** Format: U6 Zero based starting cacheline offset in to the Page Base Address

Use Global GTT (DWord 0, Bit 22):

Value	Name	Description
0h	Per Process Graphics Address	
1h	Global Graphics Address	This command will use the global GTT to translate the Address.

DWord Length (DWord 0, Bits 9:0):

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

Programming Notes	Project
The value of this field must not exceed a value 3Fh when programmed in a batch buffer with resource streamer enabled.	CHV, BSW

MI_CLFLUSH		
	5:0	**Reserved** Format: MBZ
2	31:16	**Reserved** Format: MBZ
	15:0	**Page Base Address High** Format: GraphicsAddress[47:32] This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space.
3..n	31:0	**DW Representing a Half Cache Line** Format: MBZ The information given to hardware is the DW itself, not the contents. Hardware uses the DW count of the command to determine the offset from the base to flush out. The offset is ½ cache line (8 DW = 1HW) granular so for a full page, the command will need 4096 bytes / 4 bytes per DW / 8 DW per HW = 128 DW. **Programming Notes** Always even number of "DW Representing 1/2 cacheline" terms must be programmed.

MI_CONDITIONAL_BATCH_BUFFER_END

MI_CONDITIONAL_BATCH_BUFFER_END

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

The MI_CONDITIONAL_BATCH_BUFFER_END command is used to conditionally terminate the execution of commands stored in a batch buffer initiated using a MI_BATCH_BUFFER_START command. Termination of second level batch buffer due to this command will also terminate the parent/first level batch buffer.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 36h MI_CONDITIONAL_BATCH_BUFFER_END Format: OpCode
	22	**Use Global GTT** Default Value: 0h Format: Boolean If set, this command uses the global GTT to translate the Compare Address and this command must be executing from a privileged (secure) batch buffer. If clear, the PPGTT is used to translate the Compare Address.
	21	**Compare Semaphore** Default Value: 0h Format: Boolean If set, the value from the Compare Data Dword is compared to the value from the Compare Address in memory. If the value at Compare Address is greater than the Compare Data Dword, execution of the current command buffer should continue. If clear, the parser will continue to the next command and not exit the batch buffer.
	20	**Reserved**
	19:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 2h Excludes DWord (0,1) Format: =n Total Length - 2. Excludes DWord (0,1).
1	31:0	**Compare Data Dword** Data DWord to compare to memory. The Data DWord is supplied by software to control

MI_CONDITIONAL_BATCH_BUFFER_END		
		execution of the command buffer. If the compare is enabled and the data at Semaphore Address is greater than this DWord, the execution of the command buffer should continue.
2	31:3	**Compare Address** Format: GraphicsAddress[31:3] Qword address to fetch Data Dword(DW0) from memory. HW will compare the Data Dword(DW0) with Compare Data Dword
	2:0	**Reserved** Project: All Format: MBZ
3	31:16	**Reserved** Format: MBZ
	15:0	**Compare Address High** Format: GraphicsAddress[47:32] This field specifies the 4 GB-aligned base address of GFX 4 GB virtual address space within the host's 64-bit virtual address space.

MI_CONDITIONAL_BATCH_BUFFER_END

MI_CONDITIONAL_BATCH_BUFFER_END

Project:	CHV, BSW
Source:	VideoEnhancementCS
Length Bias:	2

The MI_CONDITIONAL_BATCH_BUFFER_END command is used to conditionally terminate the execution of commands stored in a batch buffer initiated using a MI_BATCH_BUFFER_START command. Termination of second level batch buffer due to this command will also terminate the parent/first level batch buffer.

Programming Notes
This command is only valid with a 1st level batch buffer (bit 22 in MI_BATCH_BUFFER_START is set to '0')

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 36h MI_CONDITIONAL_BATCH_BUFFER_END Format: OpCode
	22	**Use Global GTT** Default Value: 0h Format: Boolean If set, this command will use the global GTT to translate the **Compare Address** and this command must be executing from a privileged (secure) batch buffer. If clear, the PPGTT will be used to translate the **Compare Address**.
	21	**Compare Semaphore** Default Value: 0h Format: Boolean If set, the value from the **Compare Data Dword** is compared to the value from the **Compare Address** in memory. If the value at **Compare Address is greater than the Compare Data Dword**, execution of current command buffer should continue. If clear, the parser will continue to the next command and not exit the batch buffer.
	20	**Reserved**
	19:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2

MI_CONDITIONAL_BATCH_BUFFER_END

		Value: 1h — Name: Excludes DWord (0,1) **[Default]** — Project: CHV, BSW
1	31:0	**Compare Data Dword** Data dword to compare memory. The Data dword is supplied by software to control execution of the command buffer. If the compare is enabled and the data at **Semaphore Address** is greater than this dword, the execution of the command buffer should continue.
2	31:3	**Compare Address** Format: GraphicsAddress[31:3] Qword address to fetch Data Dword(DW0) from memory. HW will compare the Data Dword(DW0) with Compare Data Dword
	2:0	**Reserved** Format: MBZ
3 **Project:** CHV, BSW	31:16	**Reserved** Project: CHV, BSW Format: MBZ
	15:0	**Compare Address High** Project: CHV, BSW Format: GraphicsAddress[47:32] This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space

MI_CONDITIONAL_BATCH_BUFFER_END

MI_CONDITIONAL_BATCH_BUFFER_END

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

The MI_CONDITIONAL_BATCH_BUFFER_END command is used to conditionally terminate the execution of commands stored in a batch buffer initiated using a MI_BATCH_BUFFER_START command. Termination of second level batch buffer due to this command will also terminate the parent/first level batch buffer.

Programming Notes
This command is only valid with a 1st level batch buffer (bit 22 in MI_BATCH_BUFFER_START is set to 0).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 36h MI_CONDITIONAL_BATCH_BUFFER_END Format: OpCode
	22	**Use Global GTT** Default Value: 0h DefaultVaueDesc Format: Boolean Format: U1 FormatDesc Description: If set, this command will use the global GTT to translate the Compare Address and this command must be executing from a privileged (secure) batch buffer. If clear, the PPGTT will be used to translate the Compare Address. — Project: CHV, BSW
	21	**Compare Semaphore** Default Value: 0h DefaultVaueDesc Format: Boolean If set, the value from the Compare Data Dword is compared to the value from the Compare Address in memory. If the value at Compare Address is greater than the Compare Data Dword, execution of current command buffer should continue.If clear, no comparison takes place.
	20	**Reserved**
	19:8	**Reserved**
	7:0	**DWord Length** Format: =n Total Length - 2

MI_CONDITIONAL_BATCH_BUFFER_END

DWord	Bit	Description
		<table><tr><th>Value</th><th>Name</th><th>Project</th></tr><tr><td>1h</td><td>Excludes DWord (0,1) **[Default]**</td><td>CHV, BSW</td></tr></table>
1	31:0	**Compare Data Dword** Data dword to compare memory. The Data dword is supplied by software to control execution of the command buffer. If the compare is enabled and the data at Semaphore Address is greater than this dword, the execution of the command buffer should continue.
2	31:3	**Compare Address** <table><tr><td>Format:</td><td>GraphicsAddress[31:3]</td></tr></table>Qword address to fetch compare Mask (DW0) and Data Dword(DW1) from memory. HW will do AND operation on Mask(DW0) with Data Dword(DW1) and then compare the result against Semaphore Data Dword
	2:0	**Reserved** <table><tr><td>Format:</td><td>MBZ</td></tr></table>
3 **Project:** CHV, BSW	31:16	**Reserved** <table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table>
	15:0	**Compare Address High** <table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>GraphicsAddress[47:32]</td></tr></table>This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space

MI_CONDITIONAL_BATCH_BUFFER_END

MI_CONDITIONAL_BATCH_BUFFER_END

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The MI_CONDITIONAL_BATCH_BUFFER_END command is used to conditionally terminate the execution of commands stored in a batch buffer initiated using a MI_BATCH_BUFFER_START command. Termination of second level batch buffer due to this command will also terminate the parent/first level batch buffer.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 36h MI_CONDITIONAL_BATCH_BUFFER_END Format: OpCode
	22	**Use Global GTT** Default Value: 0h If set, this command will use the global GTT to translate the **Compare Address** and this command must be executing from a privileged (secure) batch buffer. If clear, the PPGTT will be used to translate the **Compare Address**.
	21	**Compare Semaphore** Default Value: 0h If set, the value from the Compare Data Dword is compared to the value from the Compare Address in memory. If the value at Compare Address is greater than the Compare Data Dword, execution of current command buffer should continue. If clear, the parser will continue to the next command and not exit the batch buffer.
	20	**Reserved**
	19:8	**Reserved**
	7:0	**DWord Length** Format: =n Total Length - 2. Excludes DWord (0,1). Value: 1h — Name: **[Default]** — Project: CHV, BSW
1	31:0	**Compare Data Dword** Data dword to compare memory. The Data dword is supplied by software to control execution of the command buffer. If the compare is enabled and the data at Compare Address is greater than this dword, the execution of the command buffer should continue.

MI_CONDITIONAL_BATCH_BUFFER_END		
2	31:3	**Compare Address** Format: GraphicsAddress[31:3] Qword address to fetch Data Dword(DW0) from memory. HW will compare the Data Dword(DW0) with Compare Data Dword
	2:0	**Reserved** Format: MBZ
3 **Project:** CHV, BSW	31:16	**Reserved** Project: CHV, BSW
	15:0	**Compare Address High** Project: CHV, BSW Format: GraphicsAddress[47:32] This field specifies the 4GB aligned base address of Gfx 4GB virtual address space within the host's 64-bit virtual address space.

MI_COPY_MEM_MEM

MI_COPY_MEM_MEM

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

The MI_COPY_MEM_MEM command reads a DWord from memory and stores the value of that DWord to back to memory. The source and destination addresses are specified in the command. The command temporarily halts command execution.

Programming Notes
This command should not be used within a "non_privilege"batch buffer to access global virtual space, doing so will be treated as privilege access violation. Refer "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation. This command can be used within ring buffers and/or privilege batch buffers to access global virtual space.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 2Eh MI_MEM_TO_MEM Format: OpCode
	22	**Use Global GTT Source** It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear.
	21	**Use Global GTT Destination** It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear.

Use Global GTT Source values:

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.

Use Global GTT Destination values:

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.

MI_COPY_MEM_MEM		
	20:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 3 Format: =n Total Length - 2
1	31:2	**Destination Memory Address** Project: All Format: GraphicsAddress[31:2] Surface Type: MMIO Register This field specifies the address of the memory location where the value fetched specified in the DWord address above will be written. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[31:2] for a DWord register
	1:0	**Reserved**
2	31:16	**Reserved**
	15:0	**Destination Memory Address High** Project: All Format: GraphicsAddress[47:32] Surface Type: MMIO Register This field specifies the address of the memory location where the value is located that will be written to the address below. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[47:32] for a DWord register
3	31:2	**Source Memory Address** Project: All Format: GraphicsAddress[31:2] Surface Type: MMIO Register This field specifies the address of the memory location where the value is located that will be written to the address below. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[31:2] for a DWord register
	1:0	**Reserved**
4	31:16	**Reserved**
	15:0	**Source Memory Address High** Project: All Format: GraphicsAddress[47:32] Surface Type: MMIO Register This field specifies the address of the memory location where the value is located that will be written to the address below. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[47:32] for a DWord register

MI_COPY_MEM_MEM

MI_COPY_MEM_MEM	
Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The MI_COPY_MEM_MEM command reads a DWord from memory and stores the value of that DWord to back to memory. The source and destination addresses are specified in the command. The command temporarily halts command execution.

Programming Notes
This command should not be used within a "non_privilege"batch buffer to access global virtual space, doing so will be treated as privilege access violation. Refer "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation. This command can be used within ring buffers and/or privilege batch buffers to access global virtual space.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 2Eh MI_MEM_TO_MEM Format: OpCode
	22	**Use Global GTT Source** It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear.
	21	**Use Global GTT Destination** 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. This bit will determine write to memory uses Global or Per Process GTT.

Bit 22 values:

Value	Name	Description
0h	Per Process Graphics Address	
1h	Global Graphics Address	It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear.

MI_COPY_MEM_MEM

DWord	Bit	Description
		<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Per Process Graphics Address</td><td></td></tr><tr><td>1h</td><td>Global Graphics Address</td><td>This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.</td></tr></table>
	20:8	**Reserved** <table><tr><td>Format:</td><td>MBZ</td></tr></table>
	7:0	**Dword Length** <table><tr><td>Default Value:</td><td>3</td></tr><tr><td>Format:</td><td>=n Total Length - 2</td></tr></table>
1..2	63:2	**Destination Memory Address** <table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>GraphicsAddress[63:2]</td></tr></table>Surface Type: MMIO Register This field specifies the address of the memory location where the value fetched specified in the DWord address above will be written. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[63:2] for a DWord register GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].
	1:0	**Reserved** <table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table>
3..4	63:2	**Source Memory Address** <table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>GraphicsAddress[63:2]</td></tr></table>Surface Type: MMIO Register This field specifies the address of the memory location where the value is located that will be written to the address below. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[63:2] for a DWord register GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].
	1:0	**Reserved** <table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table>

MI_COPY_MEM_MEM

MI_COPY_MEM_MEM

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

The MI_COPY_MEM_MEM command reads a DWord from memory and stores the value of that DWord to back to memory. The source and destination addresses are specified in the command. The command temporarily halts command execution.

Programming Notes
This command should not be used within a "non_privilege"batch buffer to access global virtual space, doing so will be treated as privilege access violation. Refer "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation. This command can be used within ring buffers and/or privilege batch buffers to access global virtual space.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td>Command Type<table><tr><td>Default Value:</td><td>0h MI_COMMAND</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:23</td><td>MI Command Opcode<table><tr><td>Default Value:</td><td>2Eh MI_MEM_TO_MEM</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>22</td><td>Use Global GTT Source
It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Per Process Graphics Address</td><td></td></tr><tr><td>1h</td><td>Global Graphics Address</td><td>This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.</td></tr></table></td></tr>
<tr><td></td><td>21</td><td>Use Global GTT Destination
It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Per Process Graphics Address</td><td></td></tr><tr><td>1h</td><td>Global Graphics Address</td><td>This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.</td></tr></table></td></tr>
</table>

MI_COPY_MEM_MEM		
	20:8	**Reserved**
	7:0	**DWord Length** Default Value: 3 Format: =n Total Length - 2
1	31:2	**Destination Memory Address** Project: All Format: GraphicsAddress[31:2] Surface Type: MMIO Register This field specifies the address of the memory location where the value fetched specified in the DWord address above will be written. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[31:2] for a DWord register
	1:0	**Reserved**
2	31:16	**Reserved**
	15:0	**Destination Memory Address High** Project: All Format: GraphicsAddress[47:32] Surface Type: MMIO Register This field specifies the address of the memory location where the value is located that will be written to the address below. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[47:32] for a DWord register
3	31:2	**Source Memory Address** Project: All Format: GraphicsAddress[31:2] Surface Type: MMIO Register This field specifies the address of the memory location where the value is located that will be written to the address below. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[31:2] for a DWord register
	1:0	**Reserved**
4	31:16	**Reserved**
	15:0	**Source Memory Address High** Project: All Format: GraphicsAddress[47:32] Surface Type: MMIO Register This field specifies the address of the memory location where the value is located that will be written to the address below. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[47:32] for a DWord register

MI_COPY_MEM_MEM

MI_COPY_MEM_MEM

Project:	CHV, BSW
Source:	VideoEnhancementCS
Length Bias:	2

The MI_COPY_MEM_MEM command reads a DWord from memory and stores the value of that DWord to back to memory. The source and destination addresses are specified in the command. The command temporarily halts command execution.

Programming Notes
This command should not be used within a "non_privilege"batch buffer to access global virtual space, doing so will be treated as privilege access violation. Refer "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation. This command can be used within ring buffers and/or privilege batch buffers to access global virtual space.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 2Eh MI_MEM_TO_MEM Format: OpCode
	22	**Use Global GTT Source** It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear.
	21	**Use Global GTT Destination** It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear.

Use Global GTT Source:

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.

Use Global GTT Destination:

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.

MI_COPY_MEM_MEM

	20:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 3 Format: =n Total Length - 2
1	31:2	**Destination Memory Address** Project: All Format: GraphicsAddress[31:2] Surface Type: MMIO Register This field specifies the address of the memory location where the value fetched specified in the DWord address above will be written. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[31:2] for a DWord register
	1:0	**Reserved** Project: All Format: MBZ
2	31:16	**Reserved** Project: All Format: MBZ
	15:0	**Destination Memory Address High** Project: All Format: GraphicsAddress[47:32] Surface Type: MMIO Register This field specifies the address of the memory location where the value is located that will be written to the address below. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[47:32] for a DWord register
3	31:2	**Source Memory Address** Project: All Format: GraphicsAddress[31:2] Surface Type: MMIO Register This field specifies the address of the memory location where the value is located that will be written to the address below. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[31:2] for a DWord register
	1:0	**Reserved** Project: All Format: MBZ
4	31:16	**Reserved**

MI_COPY_MEM_MEM		
	15:0	**Source Memory Address High** <table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>GraphicsAddress[47:32]</td></tr></table>Surface Type: MMIO Register This field specifies the address of the memory location where the value is located that will be written to the address below. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[47:32] for a DWord register

MI_DISPLAY_FLIP

MI_DISPLAY_FLIP	
Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

The MI_DISPLAY_FLIP command is used to request a specific display plane to switch (flip) to display a new buffer. The buffer is specified with a starting address and pitch. The tiled attribute of the buffer start address is programmed as part of the packet.

The operation this command performs is also known as a "display flip request" operation - in that the flip operation itself will occur at some point in the future. This command specifies when the flip operation is to occur: either synchronously with vertical retrace to avoid tearing artifacts

Programming Notes
This command simply requests a display flip operation -- command execution then continues normally. There is no guarantee that the flip (even if asynchronous) will occur prior to subsequent commands being executed. (Note that completion of the MI_FLUSH_DW command does not guarantee that outstanding flip operations have completed). The MI_WAIT_FOR_EVENT command must be used to provide this synchronization to avoid back to back MI_DISPLAY_FLIP commands to the same display plane - by pausing command execution until a pending flip has actually completed. This synchronization can also be performed by use of the Display Flip Pending hardware status. See Display Flip Synchronization in the Device Programming Interface chapter of MI Functions.
After a display flip operation is requested, software is responsible for initiating any required synchronization with subsequent buffer clear or blitter operations. For multi-buffering (e.g., double buffering) operations, this will typically require updating SURFACE_STATE or the binding table to change the blitter (back) buffer. In addition, prior to any subsequent clear or blitter operations, software must typically ensure that the new blitter buffer is not actively being displayed. Again, the MI_WAIT_FOR_EVENT command or Display Flip Pending hardware status can be used to provide this synchronization. See Display Flip Synchronization in the Device Programming Interface chapter of MI Functions.
The display buffer command uses the X and Y offset for the tiled buffers from the Display Interface registers. Software is allowed to change the offset via the MMIO interface irrespective of the flip commands enqueued in the command stream. For tiled buffers, the display subsystem uses the X and Y offset in generation of the final request to memory. The offset is always updated on the next vblank for both Synchronous and Asynch Flips. It is not necessary to have a flip enqueued to update the X and Y offset
The display buffer command uses the linear DWord offset for the linear buffers from the Display Interface registers. Software is allowed to change the offset via the MMIO interface irrespective of the flip commands enqueued in the command stream. For linear buffers, the display subsystem uses the Dword offset in generation of the final request to memory. • For synchronous flips the offset is updated on the next vblank. It is not necessary to have a sync flip enqueued to update the DWord offset. • Linear memory does not support asynchronous flips.
Events must be unmasked in the Display Engine Render Response Mask Register (DE RRMR 0x44050) prior to

MI_DISPLAY_FLIP

waiting for them with a MI_WAIT_FOR_EVENT command, or in the case of flips or scanlines, prior to starting the flip or loading the scanline. Unmasked events will wake command streamer as they occur, so for improved power savings it is recommended to only unmask events that are required. Programming the DE RRMR register can be done through MMIO or a LOAD_REGISTER_IMMEDIATE command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 14h MI_DISPLAY_FLIP Format: OpCode
	22	**Async Flip Indicator** Format: Enable This bit should always be set if DW2 [1:0] == '01' (async flip). This field is required due to HW limitations. This bit is used by the blitter pipe while DW2 is used by the display hardware.
	21:19	**Display (Plane) Select** This field selects which display plane is to perform the flip operation. Value / Name: 0h: Display Plane A 1h: Display Plane B 2h: Display Sprite A 3h: Display Sprite B 4h: Display Plane C 5h: Display Sprite C
	18:17	**Flip Done Forward** Project: CHV, BSW This field selects the forwarding of Flip Done message to GUC by CS when execlists are enabled. Value / Name / Description: 0h: Never Forward: Never forward the message to GUC. 1h: Always Forward: Always forward the message to GUC unconditionally. 2h: Conditionally Forward: Forward the message to GUC only when the corresponding context is switched out. 3h: Reserved
	16	**Override Pipe/Plane** Project: CHV, BSW

MI_DISPLAY_FLIP

		<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>1h</td><td>Use 15:8 bits</td><td></td></tr><tr><td>0h</td><td>Use bits 21:19</td><td>("client mode")</td></tr></table>
	15:13	**Reserved** <table><tr><td>Format:</td><td>MBZ</td></tr></table>
	12:8	**Display Plane Select** <table><tr><td>Project:</td><td>CHV, BSW</td></tr></table> <table><tr><th>Value</th><th>Name</th></tr><tr><td>00000b</td><td>Display Plane A</td></tr><tr><td>00001b</td><td>Display Plane B</td></tr><tr><td>00010b</td><td>Display Plane C</td></tr><tr><td>00011b</td><td>Reserved</td></tr><tr><td>00100b</td><td>Display Sprite A</td></tr><tr><td>00101b</td><td>Display Sprite B</td></tr><tr><td>00110b</td><td>Display Sprite C</td></tr><tr><td>00111b</td><td>Display Sprite A2</td></tr><tr><td>01000b</td><td>Display Sprite B2</td></tr><tr><td>01001b</td><td>Display Sprite C2</td></tr><tr><td>01010b</td><td>Display Sprite A3</td></tr><tr><td>01011b</td><td>Display Sprite B3</td></tr><tr><td>01100b</td><td>Display Sprite C3</td></tr><tr><td>01101b</td><td>Reserved</td></tr><tr><td>01110b</td><td>Reserved</td></tr><tr><td>01111b</td><td>Reserved</td></tr><tr><td>1xxxxb</td><td>Remaining encodings reserved</td></tr></table>
	7:0	**DWord Length** <table><tr><td>Format:</td><td>=n Total Length - 2</td></tr></table> For Synchronous Flips and Asynchronous Flips, this field must be programmed to 1h for a total length of 3. <table><tr><th>Value</th><th>Name</th><th>Project</th><th>Exists If</th></tr><tr><td>0h</td><td>Excludes DWord (0,1) [Default]</td><td></td><td></td></tr><tr><td>1h</td><td></td><td></td><td>([Flip Type]!='Stereo 3D Flip')</td></tr><tr><td>2h</td><td></td><td>CHV, BSW</td><td>([Flip Type]=='Stereo 3D Flip')</td></tr></table>
1	31	**Reserved** <table><tr><td>Project:</td><td>CHV, BSW</td></tr></table>

<table>
<tr><th colspan="3">MI_DISPLAY_FLIP</th></tr>
<tr><td></td><td>30:16</td><td>Reserved<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>15:6</td><td>Reserved<table><tr><td>Project:</td><td>All</td></tr></table></td></tr>
<tr><td></td><td>5:1</td><td>Reserved<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>0</td><td>Tile Parameter<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>Enable</td></tr></table>For Asynchronous Flips, this parameter cannot be changed. All the flips in a flip chain should maintain the same tile parameter as programmed with the last synchronous flip or direct thru MMIO.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Linear [Default]</td><td>For Syncronous Flips Only</td></tr><tr><td>1h</td><td>Tiled X</td><td></td></tr></table><table><tr><th>Programming Notes</th></tr><tr><td>Performing a synchronous or asynchronous flip will drop any previous synchronous flip that has not yet completed.</td></tr></table></td></tr>
<tr><td>2</td><td>31:12</td><td>Display Buffer Base Address<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>GraphicsAddress[31:12]</td></tr></table>This field specifies Bits 31:12 of the Graphics Address of the new display buffer.<table><tr><th>Programming Notes</th></tr><tr><td>The Display buffer must reside completely in Main Memory.</td></tr><tr><td>This address is always translated via the global (rather than per-process) GTT</td></tr></table></td></tr>
<tr><td></td><td>11:3</td><td>Reserved<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>2</td><td>Reserved<table><tr><td>Project:</td><td>CHV, BSW</td></tr></table></td></tr>
<tr><td></td><td>1:0</td><td>Flip Type<table><tr><td>Project:</td><td>CHV, BSW</td></tr></table>This field specifies whether the flip operation should be performed asynchronously to vertical retrace.</td></tr>
</table>

MI_DISPLAY_FLIP

Value	Name	Description	Project
00b	Sync Flip **[Default]**	The flip will occur during the vertical blanking interval - thus avoiding any tearing artifacts.	All
01b	Async Flip	The flip will occur "as soon as possible" - and may exhibit tearing artifacts	All
1b	Reserved		All

Programming Notes

- The Display Buffer Pitch and Tile parameter cannot be changed for asynchronous flips (i.e., the new buffer must have the same pitch/tile format as the previous buffer).
- Async flips are supported on X-Tiled Frame buffers only.
- For Asynch Flips the Buffers used must be 32KB aligned.
- Asynch flips are supported on Display Planes A and B and C only.

DWord 3 — Project: CHV, BSW

31:12 Reserved

11:3 Reserved

Project:	CHV, BSW
Format:	MBZ

2 Reserved

Project:	CHV, BSW

1:0 Flip Type

Project:	CHV, BSW

This field specifies whether the flip operation should be performed asynchronously to vertical retrace.

Value	Name	Description	Project
00b	Sync Flip **[Default]**	The flip will occur during the vertical blanking interval - thus avoiding any tearing artifacts.	All
01b	Async Flip	The flip will occur "as soon as possible" - and may exhibit tearing artifacts	All

Programming Notes

- The Display Buffer Pitch and Tile parameter cannot be changed for asynchronous flips (i.e., the new buffer must have the same pitch/tile format as the previous buffer).
- Async flips are supported on X-Tiled Frame buffers only.
- For Asynch Flips the Buffers used must be 32KB aligned.
- Asynch flips are supported on Display Planes A and B and C only.

MI_FLUSH_DW

MI_FLUSH_DW

Project:	CHV, BSW
Source:	VideoEnhancementCS
Length Bias:	2

The MI_FLUSH_DW command is used to perform an internal "flush" operation. The parser pauses on an internal flush until all drawing engines have completed any pending operations. In addition, this command can also be used to:

- Flush any dirty data to memory.
- Invalidate the TLB cache inside the hardware

Usage note: After this command is completed with a Store DWord enabled, CPU access to graphics memory will be coherent (assuming the Render Cache flush is not inhibited).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND
	28:23	**MI Command Opcode** Default Value: 26h MI_FLUSH_DW
	22	**Reserved** Project: All
	21	**Store Data Index** Project: All 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).
	20:19	**Reserved**
	18	**TLB Invalidate** Project: All Format: U1

MI_FLUSH_DW

Description
If ENABLED, all TLBs belonging to Video Enhancement Engine will be invalidated once the flush operation is complete. This bit is only valid when the Post-Sync Operation field is a value of 1h or 3h.
If GFX_MODE (0x229c) bit 13, this command will cause a config write to MMIO register space with the address 0x4f100.

17 Reserved

Project:	CHV, BSW
Format:	MBZ

16 Reserved

Project:	All
Format:	MBZ

15:14 Post-Sync Operation

Project:	All

Value	Name	Description	Project
0h	No Write	No write occurs as a result of this instruction. This can be used to implement a "trap" operation, etc.	All
1h	Write Immediate Data	Write the QWord containing Immediate Data Low, High DWs to the Destination Address	All
2h	Reserved	Reserved	All
3h	Write TIMESTAMP register	Write the TIMESTAMP register to the Destination Address. The upper 28 bits of the TIMESTAMP register are tied to '0'.	

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

13:10 Reserved

Project:	All
Format:	MBZ

9 Reserved

8 Notify Enable

Project:	All
Format:	U1

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.

MI_FLUSH_DW

DWord	Bit	Description
	7	**Reserved** Project: All Format: MBZ
	6	**Reserved** Project: CHV, BSW
	5:0	**DWord Length** Project: All Format: =n Total Length - 2 Value: 3h — Name: Excludes DWord (0,1) = 2 for DWord, 3 for QWord **[Default]** — Project: CHV, BSW
1	31:3	**Address** Project: All Format: GraphicsAddress[31:3]U28 This field specifies Bits 31:3 of the Address where the DWord or QWord will be stored. Note that the address can only be QWord aligned, irrespective of data size.
	2	**Destination Address Type** Project: All Defines address space of Destination Address Value 0h — Name PPGTT — Description: Use PPGTT address space for DW write — Project: All Value 1h — Name GGTT — Description: Use GGTT address space for DW write — Project: All **Programming Notes** Ignored if "No write" is the selected in Operation.
	1:0	**Reserved** Project: All Format: MBZ
2 **Project:** CHV, BSW	31:16	**Reserved** Project: CHV, BSW Format: MBZ
	15:0	**Address High** Project: CHV, BSW Format: GraphicsAddress[47:32]U64 This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space

MI_FLUSH_DW		
3..4 **Project:** CHV, BSW	31:0	**Immediate Data** Project: CHV, BSW This field specifies the DWord value to be written to the targeted location. DW2 is the lower DW if QW is desired. Only valid when 15:14 in header is set to 1h To avoid hitting a known hardware bug, drivers cannot send a QW write when bit 5 of the address is '1'

MI_FLUSH_DW

MI_FLUSH_DW

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

The MI_FLUSH_DW command is used to perform an internal "flush" operation. The parser pauses on an internal flush until all drawing engines have completed any pending operations. In addition, this command can also be used to: Flush any dirty data to memory. Invalidate the TLB cache inside the hardware

Usage note: After this command is completed with a Store DWord enabled, CPU access to graphics memory will be coherent (assuming the Render Cache flush is not inhibited).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND
	28:23	**MI Command Opcode** Default Value: 26h MI_FLUSH_DW
	22	**Reserved** Project: All Format: U1
	21	**Store Data Index** Project: CHV, BSW Format: U1 **Description** (Project: CHV, BSW): **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).
	20:19	**Reserved** Project: All Format: MBZ
	18	**TLB Invalidate** Project: CHV, BSW Format: U1

MI_FLUSH_DW

		Description If ENABLED, all TLBs belonging to Blitter Engine will be invalidated once the flush operation is complete. This bit is only valid when the Post-Sync Operation field is a value of 1h or 3h. If GFX_MODE (0x229c) bit 13, this command will cause a config write to MMIO register space with the address 0x4f100.
	17	**Reserved** Project: CHV, BSW Format: MBZ
	16	**Reserved** Project: All Format: MBZ
	15:14	**Post-Sync Operation** Project: CHV, BSW BitFieldDesc (see table below) **Programming Notes** If executed in a non-secure batch buffer, the address given is in a PPGTT address space. If in a secure ring or batch, the address given is in GGTT space.
	13:10	**Reserved** Project: All Format: MBZ
	9	**Reserved** Project: CHV, BSW Format: MBZ
	8	**Notify Enable** Project: CHV, BSW Format: U1 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.

Post-Sync Operation BitFieldDesc:

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 QWord	Write the QWord containing Immediate Data Low, High DWs to the Destination Address
2h	Reserved	Reserved
3h	Write TIMESTAMP Register	Write the TIMESTAMP register to the Destination Address. The upper 28 bits of the TIMESTAMP register are tied to '0'.

<table>
<tr><th colspan="3">MI_FLUSH_DW</th></tr>
<tr><td></td><td>7:6</td><td>Reserved</td></tr>
<tr><td></td><td>5:0</td><td>DWord Length
<table>
<tr><td>Project:</td><td>All</td></tr>
<tr><td>Format:</td><td>=n Total Length - 2</td></tr>
</table>
<table>
<tr><th>Value</th><th>Name</th><th>Project</th></tr>
<tr><td>3h</td><td>Excludes DWord (0,1) = 2 for DWord, 3 for QWord [Default]</td><td>CHV, BSW</td></tr>
</table>
</td></tr>
<tr><td>1</td><td>31:3</td><td>Address
<table>
<tr><td>Project:</td><td>CHV, BSW</td></tr>
<tr><td>Format:</td><td>GraphicsAddress[31:3]U28</td></tr>
</table>
This field specifies Bits 31:3 of the Address where the DWord or QWord will be stored. Note that the address can only be QWord aligned, irrespective of data size.
</td></tr>
<tr><td></td><td>2</td><td>Destination Address Type
<table>
<tr><td>Project:</td><td>All</td></tr>
</table>
Defines address space of Destination Address
<table>
<tr><th>Value</th><th>Name</th><th>Description</th><th>Project</th></tr>
<tr><td>0h</td><td>PPGTT</td><td>Use PPGTT address space for DW write</td><td>All</td></tr>
<tr><td>1h</td><td>GGTT</td><td>Use GGTT address space for DW write</td><td>All</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>Ignored if "No write" is the selected in Operation.</td></tr>
</table>
</td></tr>
<tr><td></td><td>1:0</td><td>Reserved</td></tr>
<tr><td>2
Project:
CHV,
BSW</td><td>31:16</td><td>Reserved
<table>
<tr><td>Project:</td><td>CHV, BSW</td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
</table>
</td></tr>
<tr><td></td><td>15:0</td><td>Address High
<table>
<tr><td>Project:</td><td>CHV, BSW</td></tr>
<tr><td>Format:</td><td>GraphicsAddress[47:32]U64</td></tr>
</table>
This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space
</td></tr>
<tr><td>3..4
Project:
CHV,
BSW</td><td>31:0</td><td>Immediate Data
<table>
<tr><td>Project:</td><td>CHV, BSW</td></tr>
</table>
<table>
<tr><td>This field specifies the DWord value to be written to the targeted location. DW2 is the lower DW if QW is desired. Only valid when 15:14 in header is set to 1h</td></tr>
<tr><td>To avoid hitting a known hardware bug, drivers cannot send a QW write when bit 5 of the address is '1'</td></tr>
</table>
</td></tr>
</table>

MI_FLUSH_DW

MI_FLUSH_DW

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

The MI_FLUSH_DW command is used to perform an internal "flush" operation. The parser pauses on an internal flush until all drawing engines have completed any pending operations. In addition, this command can also be used to:Flush any dirty data to memory. Invalidate the TLB cache inside the hardware Usage note: After this command is completed with a Store DWord enabled, CPU access to graphics memory will be coherent (assuming the Render Cache flush is not inhibited).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND
	28:23	**MI Command Opcode** Default Value: 26h MI_FLUSH_DW
	22	**Reserved** Project: CHV, BSW
	21	**Store Data Index** Project: CHV, BSW Format: U1 Description / Project: **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). — CHV, BSW
	20:19	**Reserved** Format: MBZ
	18	**TLB Invalidate** Project: CHV, BSW Format: U1 If ENABLED, all TLBs belonging to Video Engine will be invalidated once the flush operation is complete. This bit is only valid when the Post-Sync Operation field is a value of 1h or 3h.

MI_FLUSH_DW

	17	**Reserved** Project: CHV, BSW Format: MBZ
	16	**Reserved** Format: MBZ
	15:14	**Post-Sync Operation** Project: CHV, BSW BitFieldDesc
	13:10	**Reserved** Project: All Format: MBZ
	9	**Reserved** Project: CHV, BSW Format: MBZ
	8	**Notify Enable** Project: CHV, BSW Format: U1 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	**Video Pipeline Cache invalidate** Project: CHV, BSW Format: U1 Enable the invalidation of the video cache at the end of this flush
	6	**Reserved**

Post-Sync Operation (15:14) BitFieldDesc:

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	HW implicitly detects the Data size to be Qword or Dword to be written to memory based on the command dword length programmed . When Dword Length indicates Qword, Writes the QWord containing Immediate Data Low, High DWs to the Destination Address . When Dword Length indicates Dword, Writes the DWord containing Immediate Data Low to the Destination Address
2h	Reserved	Reserved
3h		Write the TIMESTAMP register to the Destination Address. The upper 28 bits of the TIMESTAMP register are tied to '0'.

MI_FLUSH_DW

DWord	Bit	Description
		Project: CHV, BSW
	5:0	**DWord Length** Format: =n Total Length - 2 **Value** | **Name** | **Project** 3h | Excludes DWord (0,1) = 2 for DWord, 3 for QWord **[Default]** | CHV, BSW
1	31:3	**Address** Format: GraphicsAddress[31:3]U28 This field specifies Bits 31:3 of the Address where the DWord or QWord will be stored. Note that the address can only be QWord aligned, irrespective of data size.
	2	**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 the selected in Operation.
	1:0	**Reserved** Format: MBZ
2 **Project:** CHV, BSW	31:16	**Reserved** Project: CHV, BSW Format: MBZ
	15:0	**Address High** Project: CHV, BSW Format: GraphicsAddress[47:32]U64 This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space
3..4 **Project:** CHV, BSW	31:0	**Immediate Data** Project: CHV, BSW This field specifies the DWord value to be written to the targeted location. DW2 is the lower DW if QW is desired. Only valid when 15:14 in header is set to 1h To avoid hitting a known hardware bug, drivers cannot send a QW write when bit 5 of the address is '1'

MI_LOAD_REGISTER_IMM

MI_LOAD_REGISTER_IMM

Project:	CHV, BSW
Source:	VideoEnhancementCS
Length Bias:	2

The MI_LOAD_REGISTER_IMM command requests a write of up to a DWord constant supplied in the command to the specified Register Offset (i.e., offset into Memory-Mapped Register Range). The register is loaded before the next command is executed.

- The behavior of this command is controlled by Dword 3, Bit 8 (Disable Register Access) of the RINGBUF register. If this command is disallowed then the command stream converts it to a NOOP.
- If this command is executed from a batch buffer then the behavior of this command is controlled by Dword 0, Bit 8 (Security Indicator) of the BATCH_BUFFER_START Command. If the batch buffer is non-secure then the command stream converts this command to a NOOP.
- The following addresses should NOT be used for LRIs
  1. 0x8800 - 0x88FF
  2. >= 0x40000

Any offset that is to a destination outside of the GT core will allow the parser to continue once the cycle is at the GT boundry and not destination. Any other address will ensure the destination is updated prior to parsing the next command

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 22h MI_LOAD_REGISTER_IMM Format: OpCode
	22:12	**Reserved**
	11:8	**Byte Write Disables** Project: All Format: Enable[4] (bit 8 corresponds to Data DWord [7:0]). Range: Must specify a valid register write operation If [11:8] is '1111b', then the register write will not occur. If [11:8] is '0000b', then the register DW will be updated. Any other value, the behavior will be specifically specified by the register or the behavior is undefined.

MI_LOAD_REGISTER_IMM		
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Project: All Format: =n Total Length - 2
1	31:23	**Reserved** Project: All Format: MBZ
	22:2	**Register Offset** Project: All Format: MmioAddress[22:2] This field specifies bits [22:2] of the offset into the Memory Mapped Register Range (i.e., this field specifies a DWord offset).Mapped **Programming Notes** — **Project** Bits 22:18 must be zero. Setting these bits could cause a hang due to PM requesting a stop at the same time the request is going to a MMIO space outside the GT core. — CHV, BSW
	1:0	**Reserved** Project: All Format: MBZ
2	31:0	**Data DWord** Project: All Format: U32 This field specifies the DWord value to be written to the targeted location.

MI_LOAD_REGISTER_IMM

MI_LOAD_REGISTER_IMM

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

The MI_LOAD_REGISTER_IMM command requests a write of up to a DWord constant supplied in the command to the specified Register Offset (i.e., offset into Memory-Mapped Register Range). The register is loaded before the next command is executed.

Any offset that is to a destination outside of the GT core will allow the parser to continue once the cycle is at the GT boundry and not destination. Any other address will ensure the destination is updated prior to parsing the next command

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND
	28:23	**MI Command Opcode** Default Value: 22h MI_
	22:12	**Reserved** Project: All Format: MBZ
	11:8	**Byte Write Disables** Format: Enable[4] Bit 8 corresponds to Data DWord [7:0] Range: Must specify a valid register write operation If [11:8] is '1111b', then the register write will not occur. If [11:8] is '0000b', then the register DW will be updated. Any other value, the behavior will be specifically specified by the register or the behavior is undefined.
	7:0	**DWord Length** Default Value: 1h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:23	**Reserved** Format: MBZ
	22:2	**Register Offset** Format: U21 Format: MmioAddress[22:2] This field specifies bits [22:2] of the offset into the Memory Mapped Register Range (i.e., this field specifies a DWord offset).

<table>
<tr><th colspan="3">MI_LOAD_REGISTER_IMM</th></tr>
<tr><td></td><td></td><td><table><tr><th>Programming Notes</th><th>Project</th></tr><tr><td>Bits 22:18 must be zero. Setting these bits could cause a hang due to PM requesting a stop at the same time the request is going to a MMIO space outside the GT core.</td><td>CHV, BSW</td></tr></table></td></tr>
<tr><td></td><td>1:0</td><td>Reserved<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>2</td><td>31:0</td><td>Data DWord<table><tr><td>Mask:</td><td>Bytes Write Disables</td></tr><tr><td>Format:</td><td>U32</td></tr></table>This field specifies the DWord value to be written to the targeted location.</td></tr>
</table>

MI_LOAD_REGISTER_IMM

MI_LOAD_REGISTER_IMM	
Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The MI_LOAD_REGISTER_IMM command requests a write of up to a DWord constant supplied in the command to the specified Register Offset (i.e., offset into Memory-Mapped Register Range).
Any offset that is to a destination outside of the GT core will allow the parser to continue once the cycle is at the GT boundry and not destination. Any other address will ensure the destination is updated prior to parsing the next command

Programming Notes

A stalling flush must be sent down pipeline before issuing this command. The behavior of this command is controlled by Dword 3, Bit 8 (Disable Register Access) of the RINGBUF register. If this command is disallowed then the command stream converts it to a NOOP.
If this command is executed from a BB then the behavior of this command is controlled by Dword 0, Bit 8 (Security Indicator) of the BATCH_BUFFER_START Command. If the batch buffer is insecure then the command stream converts this command to a NOOP. Note that the corresponding ring buffer must allow a register update for this command to execute.
To ensure this command gets executed before upcoming commands in the ring, either a stalling pipeControl should be sent after this command, or MMIO 0x20C0 bit 7 should be set to 1.
When base address of 0x180000 is added to the Register Offset, when executed will result in updating of the register in the other GT in GTB mode of operation then the GT from which this instruction is executed. When this instruction is executed by Command Streamer with COREID-0 will result in updating the register in GT with COREID-1 and vice versa, when base address of 0x180000 is added to the register offset.
The following addresses should NOT be used for LRIs:

1. 0x8800 - 0x88FF
2. >= 0xC0000

Limited LRI cycles to the Display Engine 0x40000-0xBFFFF) are allowed, but must be spaced to allow only one pending at a time. This can be done by issuing an SRM to the same address immediately after each LRI.

Programming an MMIO register is equivalent to programming a non-pipeline state to the hardware and hence an explicit stalling flush needs to be programmed prior to programming this command. However for certain MMIO registers based on their functionality doing an explicit stalling flush is exempted. Listed below are the exempted registers.

- 3DPRIM_END_OFFSET - Auto Draw End Offset [CHV, BSW]
- 3DPRIM_START_VERTEX - Load Indirect Start Vertex [CHV, BSW]
- 3DPRIM_VERTEX_COUNT - Load Indirect Vertex Count [CHV, BSW]
- 3DPRIM_INSTANCE_COUNT - Load Indirect Instance Count [CHV, BSW]
- 3DPRIM_START_INSTANCE - Load Indirect Start Instance [CHV, BSW]
- 3DPRIM_BASE_VERTEX - Load Indirect Base Vertex [CHV, BSW]

Writes to the range 0x9400-0x97FF must be either be avoided, or serialized with a read (e.g. STORE_REGISTER_MEM) between them.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 22h MI_LOAD_REGISTER_IMM Format: OpCode
	22:13	**Reserved** Format: MBZ
	12	**Reserved** Project: CHV, BSW
	11:8	**Byte Write Disables** Format: Enable[4] Bit 8 corresponds to Data DWord [7:0] Range: Must specify a valid register write operation If [11:8] is '1111b', then this command will behave as a NOOP. Otherwise, the value is forwarded to the destination register.
	7:0	**DWord Length** Default Value: 1h Excludes DWord (0,1) Format: =n Total Length - 2. Excludes DWord (0,1).
1	31:23	**Reserved** Format: MBZ
	22:2	**Register Offset** Format: MmioAddress[22:2] This field specifies bits [22:2] of the offset into the Memory Mapped Register Range (i.e., this field specifies a DWord offset).
	1:0	**Reserved** Format: MBZ
2	31:0	**Data DWord** Mask: Bytes Write Disables Format: U32 This field specifies the DWord value to be written to the targeted location.

MI_LOAD_REGISTER_IMM

<table>
<tr><th colspan="3">MI_LOAD_REGISTER_IMM</th></tr>
<tr><td colspan="3">Project: CHV, BSW
Source: VideoCS
Length Bias: 2</td></tr>
<tr><td colspan="3">The MI_LOAD_REGISTER_IMM command requests a write of up to a DWord constant supplied in the command to the specified Register Offset (i.e., offset into Memory-Mapped Register Range). The register is loaded before the next command is executed.
Any offset that is to a destination outside of the GT core will allow the parser to continue once the cycle is at the GT boundry and not destination. Any other address will ensure the destination is updated prior to parsing the next command</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="5">0</td><td>31:29</td><td>Command Type
Default Value: 0h MI_COMMAND
Format: OpCode</td></tr>
<tr><td>28:23</td><td>MI Command Opcode
Default Value: 22h MI_LOAD_REGISTER_IMM
Format: OpCode</td></tr>
<tr><td>22:12</td><td>Reserved</td></tr>
<tr><td>11:8</td><td>Byte Write Disables
Format: Enable[4] (bit 8 corresponds to Data DWord [7:0]).

Range: Must specify a valid register write operation
If [11:8] is '1111b', then the register write will not occur. If [11:8] is '0000b', then the register DW will be updated. Any other value, the behavior will be specifically specified by the register or the behavior is undefined.</td></tr>
<tr><td>7:0</td><td>DWord Length
Default Value: 0h Excludes DWord (0,1)
Format: =n Total Length - 2</td></tr>
<tr><td rowspan="2">1</td><td>31:23</td><td>Reserved</td></tr>
<tr><td>22:2</td><td>Register Offset
Format: MmioAddress[22:2]
This field specifies bits [22:2] of the offset into the Memory Mapped Register Range (i.e., this field specifies a DWord offset).Mapped
<table><tr><th>Programming Notes</th><th>Project</th></tr><tr><td>Bits 22:18 must be zero. Setting these bits could cause a hang due to PM requesting a stop at the same time the request is going to a MMIO space outside the GT core.</td><td>CHV, BSW</td></tr></table></td></tr>
</table>

MI_LOAD_REGISTER_IMM		
	1:0	**Reserved** Format: MBZ
2	31:0	**Data DWord** Format: U32 FormatDesc This field specifies the DWord value to be written to the targeted location.

MI_LOAD_REGISTER_MEM

MI_LOAD_REGISTER_MEM

Project:	CHV, BSW
Source:	RenderCS, BlitterCS, VideoCS, VideoEnhancementCS
Length Bias:	2

The MI_LOAD_REGISTER_MEM command requests from a memory location and stores that DWord to a register.

Programming Notes
The command temporarily halts commands that will cause cycles down the 3D pipeline.
The following addresses should NOT be used for MMIO writes: • 0x8800 - 0x88FF • >= 0xC0000 Limited MMIO writes cycles to the Display Engine 0x40000-0xBFFFF) are allowed, but must be spaced to allow only one pending at a time. This can be done by issuing an SRM to the same address immediately after each MMIO write.
Any updates to the memory location exercised by this command must be ensured to be coherent in memory prior to programming of this command. This must be achieved by programming MI_ATOMIC (write to scratch space) with "CS STALL" set prior to programming of this command. Example: MI_STORE_REGISTE_MEM (0x2288, 0x2CF0_0000) ……… ……… MI_ATOMIC (MOV, Dummy data, Scratch Address) MI_LOAD_REGISTER_MEM(0x2288, 0x2CF0_0000)
This command should not be used within a non-privilege batch buffer to access global virtual space, doing so will be treated as privilege access violation. Refer "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation.
This command is not allowed to update the privilege register range when executed from a non-privilege batch buffer.
Writes to the range 0x9400-0x97FF must be either be avoided, or serialized with a read (e.g. STORE_REGISTER_MEM) between them.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 29h MI_LOAD_REGISTER_MEM Format: OpCode
	22	**Use Global GTT** Format: Boolean This bit if set when executing from a non-privileged batch buffer will be treated as privilege access violation. It is allowed for this bit to be clear when executing this command from a

MI_LOAD_REGISTER_MEM

		privileged (secure) batch buffer or ring buffer. This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.
	21	**Async Mode Enable** If this bit is set then the command stream will not wait for completion of this command before executing the next command. Please refer to the LOAD_INDIRECT and Predicate registers for usage of this bit.
	20	**Reserved** Project:
	19:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2. Excludes DWord (0,1). Value | Name | Project 2h | Excludes DWord (0,1) **[Default]** | CHV, BSW
1	31:23	**Reserved** Format: MBZ
	22:2	**Register Address** Format: MMIOAddress[22:2] This field specifies Bits 22:2 of the Register offset the DWord will be written to. As the register address must be DWord-aligned, Bits 1:0 of that address MBZ. Programming Notes | Project | Source Bits 22:18 must be zero. Setting these bits could cause a hang due to PM requesting a stop at the same time the request is going to a MMIO space outside the GT core. | CHV, BSW | BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS
	1:0	**Reserved** Format: MBZ
2..3 **Project:** CHV, BSW	63:2	**Memory Address** Project: CHV, BSW Format: GraphicsAddress[63:2] This field specifies the address of the memory location where the register value specified in the DWord above will read from. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[63:2] for a DWord register GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].
	1:0	**Reserved**

MI_LOAD_REGISTER_REG

MI_LOAD_REGISTER_REG

Project:	CHV, BSW
Source:	CommandStreamer
Length Bias:	2

The MI_LOAD_REGISTER_REG command reads from a source register location and writes that value to a destination register location.

Any offset that is to a destination outside of the GT core will allow the parser to continue once the cycle is at the GT boundry and not destination. Any other address will ensure the destination is updated prior to parsing the next command

Programming Notes
The command temporarily halts commands that will cause cycles down the 3D pipeline.
Destination register with mask implemented will not get updated unless the value read from source register has the bits corresponding to the mask bits set. Note that any mask implemented register when read returns "0" for the bits corresponding to mask location. When the source and destination are mask implemented registers, destination register will not get updated with the source register contents.
This command is not allowed to update the privilege register range when executed from a non-privilege batch buffer.
Writes to the range 0x9400-0x97FF must be either be avoided, or serialized with a read (e.g. STORE_REGISTER_MEM) between them.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 2Ah Format: OpCode
	22:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 1h Format: =n Total Length - 2. Excludes DWord (0,1).
1	31:23	**Reserved**
	22:2	**Source Register Address** Format: MMIOAddress[22:2]MMIO_Register This field specifies Bits 22:2 of the Register offset the DWord will be written to. As the register address must be DWord-aligned, Bits 1:0 of that address MBZ.

MI_LOAD_REGISTER_REG

		Programming Notes / **Project** / **Source** Bits 22:18 must be zero. Setting these bits could cause a hang due to PM requesting a stop at the same time the request is going to a MMIO space outside the GT core. / CHV, BSW / BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS
	1:0	**Reserved** Format: MBZ
2	31:23	**Reserved** Format: MBZ
	22:2	**Destination Register Address** Format: MMIOAddress[22:2]MMIO_Register This field specifies Bits 22:2 of the Register offset the DWord will be written to. As the register address must be DWord-aligned, Bits 1:0 of that address MBZ. **Programming Notes** / **Project** / **Source** Bits 22:18 must be zero. Setting these bits could cause a hang due to PM requesting a stop at the same time the request is going to a MMIO space outside the GT core. / CHV, BSW / BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS
	1:0	**Reserved** Format: MBZ

MI_LOAD_SCAN_LINES_EXCL

MI_LOAD_SCAN_LINES_EXCL

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

The MI_LOAD_SCAN_LINES_EXCL command is used to initialize the Scan Line Window registers for a specific Display Pipe. If the display refresh is *outside* this window the Display Engine asserts a signal that is used by the command parser to process the WAIT_FOR_EVENT command (i.e., the parser will wait while outside). This command overrides the Scan Line Window defined by any previous MI_LOAD_SCAN_LINES_INCL or MI_LOAD_SCAN_LINES_EXCL commands targeting the specific display pipe.
Note: The two scan-line numbers are inclusive. If programmed to the same values, that single line defines the region in question.
Always place an even number of MI_LOAD_SCAN_LINES_EXCL/INCL at a time into the ring buffer. If only a single MI_LOAD_SCAN_LINES_EXCL/INCL is desired, just add a second identical MI_LOAD_SCAN_LINES_EXCL/INCL command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 13h MI_LOAD_SCAN_LINES_EXCL Format: OpCode
	22	**Reserved** Project: All Format: MBZ
	21:19	**Display Pipe Select** Project: All Format: U3 This field selects which Display Engine (pipe) this command is targeting. Value / Name / Project: 0h / Display Pipe A / All 1h / Display Pipe B / All 4h / Display Pipe C / All
	18:17	**Reserved**
	16:6	**Reserved**
	5:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2

MI_LOAD_SCAN_LINES_EXCL		
1	31:16	**Start Scan Line Number** Format: U16 In scan lines, where scan line 0 is the first line of the display frame. This field specifies the starting scan line number of the Scan Line Window. Range: [0, Display Buffer height in lines-1]
	15:0	**End Scan Line Number** Format: U16 In scan lines, where scan line 0 is the first line of the display frame. This field specifies the ending scan line number of the Scan Line Window. Range: [0, Display Buffer height in lines-1]

MI_LOAD_SCAN_LINES_EXCL

MI_LOAD_SCAN_LINES_EXCL

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The MI_LOAD_SCAN_LINES_EXCL command is used to initialize the Scan Line Window registers for a specific Display Pipe. If the display refresh is **outside** this window the Display Engine asserts a signal that is used by the command parser to process the WAIT_FOR_EVENT command (i.e., the parser will wait while outside). This command overrides the Scan Line Window defined by any previous MI_LOAD_SCAN_LINES_INCL or MI_LOAD_SCAN_LINES_EXCL commands targeting the specific display pipe. **Note:** The two scan-line numbers are inclusive. If programmed to the same values, that single line defines the region in question. Always place an even number of MI_LOAD_SCAN_LINES_EXCL/INCL at a time into the ring buffer. If only a single MI_LOAD_SCAN_LINES_EXCL/INCL is desired, just add a second identical MI_LOAD_SCAN_LINES_EXCL/INCL command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 13h MI_LOAD_SCAN_LINES_EXCL Format: OpCode
	22	**Reserved**
	21:19	**Display (Plane) Select** Format: U3 This field selects which display plane is to perform the scanline operation. Value / Name: 0h: Display Plane A 1h: Display Plane B 2h: Reserved 3h: Reserved 4h: Display Plane C 5h: Reserved
	18:17	**Reserved**
	16:6	**Reserved**
	5:0	**DWord Length** Default Value: 0h Format: =n Total Length - 2. Excludes DWord (0,1).

MI_LOAD_SCAN_LINES_EXCL

1	31:29	**Reserved** Format: MBZ
	28:16	**Start Scan Line Number** Format: U13 In scan lines, where scan line 0 is the first line of the display frame. Range: [0, Display Buffer height in lines-1] This field specifies the starting scan line number of the Scan Line Window.
	15:13	**Reserved** Format: MBZ
	12:0	**End Scan Line Number** Format: U13 In scan lines, where scan line 0 is the first line of the display frame. This field specifies the ending scan line number of the Scan Line Window. Range: [0, Display Buffer height in lines-1]

MI_LOAD_SCAN_LINES_INCL

MI_LOAD_SCAN_LINES_INCL	
Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

The MI_LOAD_SCAN_LINES_INCL command is used to initialize the Scan Line Window registers for a specific Display Engine. If the display refresh is *within* this window the Display Engine asserts a signal that is used by the command parser to process the WAIT_FOR_EVENT command (i.e., the parser will wait while inside of the window). This command overrides the Scan Line Window defined by any previous MI_LOAD_SCAN_LINES_INCL or MI_LOAD_SCAN_LINES_EXCL commands targeting the specific display.

Always place an even number of MI_LOAD_SCAN_LINES_EXCL/INCL at a time into the ring buffer. If only a single MI_LOAD_SCAN_LINES_EXCL/INCL is desired, just add a second identical

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 12h MI_LOAD_SCAN_LINES_INCL Format: OpCode
	22	**Reserved** Project: All Format: MBZ
	21:19	**Display Pipe Select** Project: All Format: U3 This field selects which Display Engine (pipe) this command is targeting. Value / Name: 0h: Display Pipe A 1h: Display Pipe B 4h: Display Pipe C
	18:17	**Reserved**
	16:6	**Reserved** Project: CHV, BSW Format: MBZ
	5:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2

MI_LOAD_SCAN_LINES_INCL		
1	31:16	**Start Scan Line Number** Format: U16 In scan lines, where scan line 0 is the first line of the display frame. This field specifies the starting scan line number of the Scan Line Window. Range: [0, Display Buffer height in lines-1]
	15:0	**End Scan Line Number** Format: U16 In scan lines, where scan line 0 is the first line of the display frame. This field specifies the ending scan line number of the Scan Line Window. Range: [0, Display Buffer height in lines-1]

MI_LOAD_SCAN_LINES_INCL

MI_LOAD_SCAN_LINES_INCL

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The MI_LOAD_SCAN_LINES_INCL command is used to initialize the Scan Line Window registers for a specific Display Engine. If the display refresh is **within** this window the Display Engine asserts a signal that is used by the command parser to process the WAIT_FOR_EVENT command (i.e., the parser will wait while inside the window). This command overrides the Scan Line Window defined by any previous MI_LOAD_SCAN_LINES_INCL or MI_LOAD_SCAN_LINES_EXCL commands targeting the specific display.

Always place an even number of MI_LOAD_SCAN_LINES_EXCL/INCL at a time into the ring buffer. If only a single MI_LOAD_SCAN_LINES_EXCL/INCL is desired, just add a second identical MI_LOAD_SCAN_LINES_EXCL/INCL command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 12h MI_LOAD_SCAN_LINES_INCL Format: OpCode
	22	**Reserved** Format: MBZ
	21:19	**Display (Plane) Select** Project: CHV, BSW Format: U3 This field selects which display plane is to perform the scanline operation. Value / Name: 0h: Display Plane A 1h: Display Plane B 2h: Reserved 3h: Reserved 4h: Display Plane C 5h: Reserved
	18:17	**Scan Line Event Done Forward** Project: CHV, BSW Format: Enable This field selects the forwarding of Scan Line Event Done message to GUC by CS when execlists are enabled.

MI_LOAD_SCAN_LINES_INCL

		Value | **Name** | **Description** 0h | Never Forward | Never forward the message to GUC. 1h | Always Forward | Always forward the message to GUC unconditionally. 2h | Conditionally Forward | Forward the message to GUC only when the corresponding context is switched out. 3h | Reserved |
	16:6	**Reserved** Project: CHV, BSW Format: MBZ
	5:0	**DWord Length** Default Value: 0h Format: =n Total Length - 2. Excludes DWord (0,1).
1	31	**Reserved** Project: CHV, BSW Format: Must Be One
	30	**Rserved** Project: CHV, BSW Format: MBZ
	29	**Reserved** Format: MBZ
	28:16	**Start Scan Line Number** Format: U13 In scan lines, where scan line 0 is the first line of the display frame. Range: [0, Display Buffer height in lines-1] This field specifies the starting scan line number of the Scan Line window.
	15:13	**Reserved** Format: MBZ
	12:0	**End Scan Line Number** Format: U13 In scan lines, where scan line 0 is the first line of the display frame. Range: [0, Display Buffer height in lines-1] This field specifies the ending scan line number of the Scan Line Window.

MI_LOAD_URB_MEM

MI_LOAD_URB_MEM

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The MI_LOAD_URB_MEM command requests from a memory location and stores that DWord to the URB.

Programming Notes
The command temporarily halts commands that will cause cycles down the 3D pipeline.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 2Ch MI_LOAD_URB_MEM Format: OpCode
	22:8	**Reserved**
	7:0	**DWord Length** Format: =n Total Length - 2. Excludes DWord (0,1). Value / Name / Project: 2h / **[Default]** / CHV, BSW
1	31:15	**Reserved** Format: MBZ
	14:2	**URB Address** This field specifies Bits 14:2 of the URB offset the DWord will be written in the URB. This command only supports writing below 32KB of the URB space.
	1:0	**Reserved** Format: MBZ
2..3 **Project:** CHV, BSW	63:6	**Memory Address** Project: CHV, BSW Format: GraphicsAddress[63:6] This field specifies the address of the location of where the value will be read from memory. The value must be in the first DW location of the cache line. Range = GraphicsVirtualAddress[47:6] GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].
	5:0	**Reserved**

MI_MATH

MI_MATH		
Project:	CHV, BSW	
Source:	BlitterCS	
Length Bias:	2	

The MI_MATH command allows software to send instructions to the ALU in the Command Streamer.
This command is the means by which the ALU is accessed. ALU instructions form the data payload of the MI_MATH command. An ALU instruction takes one DWord in size. The MI_MATH DWord Length is programmed based on the number of ALU instructions included, limited only by the max DWord Length supported.
When the command streamer parses an MI_MATH command, it sends the included ALU instructions to the ALU. The ALU processes any instruction in a single clock. See the ALU section for more details.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 1Ah MI_MATH Format: OpCode
	22:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Format: =n Total Length - 2. Excludes DWord (0,1).
1	31:0	**ALU INSTRUCTION 1** Format: Table Entry
2	31:0	**ALU INSTRUCTION 2** Format: Table Entry
3..n	31:0	**ALU INSTRUCTION n** Format: Table Entry

MI_MATH

MI_MATH		
Project:	CHV, BSW	
Source:	VideoCS	
Length Bias:	2	

The MI_MATH command allows software to send instructions to the ALU in the Command Streamer.
This command is the means by which the ALU is accessed. ALU instructions form the data payload of the MI_MATH command. An ALU instruction takes one DWord in size. The MI_MATH DWord Length is programmed based on the number of ALU instructions included, limited only by the max DWord Length supported.
When the command streamer parses an MI_MATH command, it sends the included ALU instructions to the ALU. The ALU processes any instruction in a single clock. See the ALU section for more details.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 1Ah MI_MATH Format: OpCode
	22:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Format: =n Total Length - 2. Excludes DWord (0,1).
1	31:0	**ALU INSTRUCTION 1** Format: Table Entry
2	31:0	**ALU INSTRUCTION 2** Format: Table Entry
3..n	31:0	**ALU INSTRUCTION n** Format: Table Entry

MI_MATH

<table>
<tr><th colspan="3">MI_MATH</th></tr>
<tr><td colspan="3">Project: CHV, BSW
Source: VideoEnhancementCS
Length Bias: 2</td></tr>
<tr><td colspan="3">The MI_MATH command allows software to send instructions to the ALU in the Command Streamer.
This command is the means by which the ALU is accessed. ALU instructions form the data payload of the MI_MATH command. An ALU instruction takes one DWord in size. The MI_MATH DWord Length is programmed based on the number of ALU instructions included, limited only by the max DWord Length supported.
When the command streamer parses an MI_MATH command, it sends the included ALU instructions to the ALU. The ALU processes any instruction in a single clock. See the ALU section for more details.</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="4">0</td><td>31:29</td><td>Command Type
Default Value: 0h MI_COMMAND
Format: OpCode</td></tr>
<tr><td>28:23</td><td>MI Command Opcode
Default Value: 1Ah MI_MATH
Format: OpCode</td></tr>
<tr><td>22:8</td><td>Reserved
Format: MBZ</td></tr>
<tr><td>7:0</td><td>DWord Length
Default Value: 0h
Format: =n Total Length - 2. Excludes DWord (0,1).</td></tr>
<tr><td>1</td><td>31:0</td><td>ALU INSTRUCTION 1
Format: Table Entry</td></tr>
<tr><td>2</td><td>31:0</td><td>ALU INSTRUCTION 2
Format: Table Entry</td></tr>
<tr><td>3..n</td><td>31:0</td><td>ALU INSTRUCTION n
Format: Table Entry</td></tr>
</table>

MI_MATH

MI_MATH	
Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The MI_MATH command allows SW to send instruction to ALU in Render Command Streamer.
MI_MATH command is the means by which ALU can be accessed. ALU instructions form the data payload of MI_MATH command, ALU instruction is dword in size. MI_MATH Dword Length should be programmed based on the number of ALU instruction packed, max number is limited by the max Dword Length supported. When MI_MATH command is parsed by command streamer it outputs the payload dwords (ALU instructions) to the ALU. ALU takes single clock to process any given instruction. Refer to B-spec "Command Streamer (CS) ALU Programming" section in Command Streamer Programming.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 1Ah MI_MATH Format: OpCode
	22:8	**Reserved** Format: MBZ
	7:6	**Reserved** Project: CHV, BSW Format: MBZ
	5:0	**DWord Length** Default Value: 0h Project: CHV, BSW Format: =n Total Length - 2. Excludes DWord (0,1).
1	31:0	**ALU INSTRUCTION 1** Format: Table Entry
2	31:0	**ALU INSTRUCTION 2** Format: Table Entry
3..n	31:0	**ALU INSTRUCTION n** Format: Table Entry

MI_NOOP

MI_NOOP	
Project:	CHV, BSW
Source:	VideoEnhancementCS
Length Bias:	1

The MI_NOOP command basically performs a "no operation" in the command stream and is typically used to pad the command stream (e.g., in order to pad out a batch buffer to a QWord boundary). However, there is one minor (optional) function this command can perform - a 22-bit value can be loaded into the MI NOPID register. This provides a general-purpose command stream tagging ("breadcrumb") mechanism (e.g., to provide sequencing information for a subsequent breakpoint interrupt).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 00h MI_NOOP Format: OpCode
	22	**Identification Number Register Write Enable** Format: Enable This field enables the value in the Identification Number field to be written into the MI NOPID register. If disabled, that register is unmodified - making this command an effective "no operation" function. Value / Name / Description: 1 / / Write th NOP_ID Register 0 / / Do not write the NOP_ID register
	21:0	**Identification Number** Project: All Format: U22 This field contains a 22-bit number which can be written to the MI NOPID register.

MI_NOOP

MI_NOOP		
Project:	CHV, BSW	
Source:	BlitterCS	
Length Bias:	1	

The MI_NOOP command basically performs a "no operation" in the command stream and is typically used to pad the command stream (e.g., in order to pad out a batch buffer to a QWord boundary). However, there is one minor (optional) function this command can perform - a 22-bit value can be loaded into the MI NOPID register. This provides a general-purpose command stream tagging ("breadcrumb") mechanism (e.g., to provide sequencing information for a subsequent breakpoint interrupt).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND
	28:23	**MI Command Opcode** Default Value: 0h MI_NOOP
	22	**Identification Number Register Write Enable** Project: All Format: Enable This field enables the value in the Identification Number field to be written into the MI NOPID register. If disabled, that register is unmodified - making this command an effective "no operation" function. Value | Name | Description | Project 0h | Disable | Do not write the NOP_ID register. | All 1h | Enable | Write the NOP_ID register. | All
	21:0	**Identification Number** Project: All Format: U22 This field contains a 22-bit number which can be written to the MI NOPID register.

MI_NOOP

<table>
<tr><th colspan="3">MI_NOOP</th></tr>
<tr><td colspan="3">Project: CHV, BSW
Source: RenderCS
Length Bias: 1</td></tr>
<tr><td colspan="3">The MI_NOOP command basically performs a "no operation" in the command stream and is typically used to pad the command stream (e.g., in order to pad out a batch buffer to a QWord boundary). However, there is one minor (optional) function this command can perform - a 22-bit value can be loaded into the MI NOPID register. This provides a general-purpose command stream tagging ("breadcrumb") mechanism (e.g., to provide sequencing information for a subsequent breakpoint interrupt).</td></tr>
<tr><td colspan="3"><table><tr><th>Performance</th><th>Project</th></tr><tr><td>The MI_NOOP process time is reduced to 1 clock. An example use of the improved NOOP throughput is for some multi-pass media applications where some unwanted media object commands are replaced by MI_NOOP commands without repacking the commands in a batch buffer.</td><td>CHV, BSW</td></tr></table></td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="4">0</td><td>31:29</td><td>Command Type<table><tr><td>Default Value:</td><td>0h MI_COMMAND</td></tr></table></td></tr>
<tr><td>28:23</td><td>MI Command Opcode<table><tr><td>Default Value:</td><td>0h MI_NOOP</td></tr></table></td></tr>
<tr><td>22</td><td>Identification Number Register Write Enable<table><tr><td>Format:</td><td>Enable</td></tr></table>This field enables the value in the Identification Number field to be written into the MI NOPID register. If disabled, that register is unmodified, making this command an effective "no operation" function.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Disable</td><td>Do not write the NOP_ID register.</td></tr><tr><td>1h</td><td>Enable</td><td>Write the NOP_ID register.</td></tr></table></td></tr>
<tr><td>21:0</td><td>Identification Number<table><tr><td>Format:</td><td>U22</td></tr></table>This field contains a 22-bit number which can be written to the MI NOPID register.</td></tr>
</table>

MI_NOOP

MI_NOOP

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	1

The MI_NOOP command basically performs a "no operation" in the command stream and is typically used to pad the command stream (e.g., in order to pad out a batch buffer to a QWord boundary). However, there is one minor (optional) function this command can perform - a 22-bit value can be loaded into the MI NOPID register. This provides a general-purpose command stream tagging ("breadcrumb") mechanism (e.g., to provide sequencing information for a subsequent breakpoint interrupt).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 00h MI_NOOP Format: OpCode
	22	**Identification Number Register Write Enable** Format: Enable This field enables the value in the Identification Number field to be written into the MI NOPID register. If disabled, that register is unmodified - making this command an effective "no operation" function. Value: 1 — Name: Write the NOP_ID register.
	21:0	**Identification Number** Format: U22 This field contains a 22-bit number which can be written to the MI NOPID register.

MI_PREDICATE

MI_PREDICATE

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	1

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 0Ch MI_PREDICATE Format: OpCode
	22:8	**Reserved** Format: MBZ
	7:6	**Load Operation** This field controls if/how the Predicate state bit is modified. Value / Name / Description: 0h / KEEP / The Predicate state bit is unmodified. 1h / Reserved / 2h / LOAD / The Predicate state bit is loaded with the combine operation result. 3h / LOADINV / The Predicate state bit is loaded with the inverted combine operation result.
	5	**Reserved** Format: MBZ
	4:3	**Combine Operation** This field controls if/how the result of the compare operation is combined with the current Predicate state bit. Value / Name / Description: 0h / SET / The combine operation output the compare result unmodified. 1h / AND / The combine operation outputs the AND of the compare result and the current Predicate state bit. 2h / OR / The combine operation outputs the OR of the compare result and the current Predicate state bit. 3h / XOR / The combine operation outputs the XOR of the compare result and the current Predicate state bit.
	2	**Reserved** Format: MBZ

MI_PREDICATE		
	1:0	**Compare Operation** This field controls how Data DWord 0 and Data DWord 1 fields are used to generate a compare operation result and possibly modify the PredicateData register.

Value	Name	Description
0h	TRUE	The compare operation outputs TRUE. The PredicateData register is unmodified.
1h	FALSE	The compare operation outputs FALSE. The PredicateData register is unmodified.
2h	SRCS_EQUAL	(MItemp0 - MItemp1) is computed and loaded into the PredicateData register. The compare operation outputs (MItemp0 == MItemp1).
3h	DELTAS_EQUAL	(MItemp0 - MItemp1) is computed and compared to the PredicateData register. If the values are equal, the compare operation outputs TRUE, otherwise it outputs FALSE. The PredicateData register is unmodified.

MI_REPORT_HEAD

MI_REPORT_HEAD	
Project:	CHV, BSW
Source:	VideoEnhancementCS
Length Bias:	1

The MI_REPORT_HEAD command causes the Head Pointer value of the ring buffer to be written to a cacheable (snooped) system memory location.
When the **Per-Process Virtual Address Space and Execlist Enable bit** is reset: The location written is relative to the address programmed in the Hardware Status Page Address Register. When the **Execlist Enable** is set, the head pointer will be reported to the PP HW Status Page.

Programming Notes
This command must not be executed from a Batch Buffer (Refer to the description of the HWS_PGA register).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 07h MI_REPORT_HEAD Format: OpCode
	22:0	**Reserved** Project: All Format: MBZ

MI_REPORT_HEAD

MI_REPORT_HEAD	
Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	1

The MI_REPORT_HEAD command causes the Head Pointer value of the active ring buffer to be written to a cacheable (snooped) system memory location.
When the **Execlist Enable** bit is reset:
The location written is relative to the address programmed in the Hardware Status Page Address Register.

Programming Notes
This command must not be executed from a Batch Buffer (Refer to the description of the HWS_PGA register). When the **Execlist Disable** is clear, the head pointer will be reported to the PP HW Status Page.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND
	28:23	**MI Command Opcode** Default Value: 07h MI_REPORT_HEAD
	22:0	**Reserved** Project: All Format: MBZ

MI_REPORT_HEAD

MI_REPORT_HEAD

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	1

The MI_REPORT_HEAD command causes the Head Pointer value of the active ring buffer to be written to a cacheable (snooped) system memory location. When Execlist Enable is set, the head pointer will be reported to the PP HW Status Page. The location written is relative to the address programmed in the Hardware Status Page Address Register.

Programming Notes
This command must not be executed from a Batch Buffer. (Refer to the description of the HWS_PGA register.)

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 07h MI_REPORT_HEAD Format: OpCode
	22:0	**Reserved** Format: MBZ

MI_REPORT_HEAD

MI_REPORT_HEAD

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	1

The MI_REPORT_HEAD command causes the Head Pointer value of the ring buffer to be written to a cacheable (snooped) system memory location.When the Per-Process Virtual Address Space and Execlist Enable bitis reset:The location written is relative to the address programmed in the Hardware Status Page Address Register. When the Execlist Enable is set, the head pointer will be reported to the PP HW Status Page.

Programming Notes
This command must not be executed from a Batch Buffer (Refer to the description of the HWS_PGA register).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 07h MI_REPORT_HEAD Format: OpCode
	22:0	**Reserved** Format: MBZ

MI_REPORT_PERF_COUNT

MI_REPORT_PERF_COUNT

Project:	BDW
Source:	RenderCS
Length Bias:	2

The MI_REPORT_PERF_COUNT command causes the GFX hardware to write out a snap-shot of performance counters to the address specified in this command along with constant ID field supplied and the time-stamp counter. This write is required to be treated as a cacheable write irrespective of GTT entry memory type. This command is specific to the render engine.

Programming Notes
This command is to be used for performance debug mode and can be inserted after events of interest (frequently before and after a 3DPRIMITIVE command). SW is entirely responsible for managing the ID field and addresses used by such a series of commands.
Graphics Memory address used by MI_REPORT_PERF_COUNT should be below 2G, i.e. bits[47:32] of the memory address must be always set to 0x0.
GTT_SELECT must not be set to 1 (i.e. GGTT) when MI_REPORT_PERF_COUNT command is programmed in a non-privileged batch buffer. Refer to the "User Mode Privileged commands" Table in MI_BATCH_BUFFER_START command section for more details. All batch buffers in PPGTT are considered as Non-privileged.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 28h MI_REPORT_PERF_COUNT Format: OpCode
	22:6	**Reserved** Format: MBZ
	5:0	**DWord Length** Format: =n Total Length - 2 Value: 2h — Name: Excludes DWord (0,1) **[Default]**

MI_REPORT_PERF_COUNT		
1..2	63:6	**Memory Address** Format: GraphicsAddress[63:6] This field specifies 64B aligned GFX MEM address where the chap counter values are reported. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47] **Programming Notes** This field is ignored if "Report to OABUFFER" bit is set.
	5	**Reserved** Format: MBZ
	4	**Core Mode Enable** Format: U1 This bit is set then the address will be offset by the Core ID:If Core ID 0, then there is no offsetIf Core ID 1, then the Memory is offset by the size of the data(64b).
	3:1	**Reserved** Format: MBZ
	0	**Use Global GTT** Format: Boolean This field when set (i.e. bit = 1) selects the GGTT for address translation. When this bit is 0 (default value), HW should use PGTT for address translation.
3	31:0	**Report ID** Format: U32 This field specifies the ID provided by SW for a given report command. It can be tracked to use different flavors of these reports based on where in command-stream they are inserted. This field is reported only when Counter Select Field is 0. **Programming Notes** If a privilege access violation occurs, the REPORT ID field in the report generated by the next legitimate MI_REPORT_PERF_COUNT will be corrupted.

MI_RS_CONTEXT

MI_RS_CONTEXT

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	1

The MI_RS_CONTEXT command is used to force a resource streamer context save or restore.

Programming Notes	Project
This command must not be used/programmed in Execution List mode of scheduling.	CHV, BSW

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 0Fh MI_RS_CONTEXT Format: OpCode
	22:1	**Reserved** Format: MBZ
	0	**Resource Streamer Save** Format: U1 This bit specifies whether the MI_RS_CONTEXT command will cause the resource streamer context to be saved or restored.

Value	Name	Description
0h	Restore	Resource Streamer context is restored
1h	Save	Resource Streamer context is saved

MI_RS_CONTROL

MI_RS_CONTROL

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	1

The MI_RS_CONTROL command is used to start or stop the Resource Streamer.

Programming Notes

- This command must be programmed only inside a Resource Streamer enabled batch buffer.
- This command provides means to selectively disable or enable Resource Streamer for set of commands in a Resource Streamer enabled batch buffer
- On re-enabling the Resource Streamer through this command, command streamer will start Resource Streamer on the next non-sync command of the batch buffer.
- This command status is render context save/restored during context switching.
- The scope of MI_RS_CONTROL is within the batch buffer it is programmed, it doesn't get carried to the following chained batch buffer or second level batch buffer. RS control status goes back to default mode of Resource Streamer Enabled on all batch buffer arbitration boundaries. Batch buffer arbitration boundaries includes calling a chained or a second level batch buffer through MI_BATCH_BUFFER_START command or terminating a batch buffer through MI_BATCH_BUFFER_END command.
- Example:

1. MI_BATCH_START (Primary batch buffer with RS enable)
2. Command 1 --> CS starts RS
3. Command 2
4. :
5. MI_RS_CONTROL (stop option) -> RS will stop on this command, CS sets RS control status to STOP.
6. Command 3
7. MI_BATCH_START (2nd level batch with RS enable not set, RS control status gets reset to default status of START)
8. :
9. MI_BATCH_END (Second Level Batch End)
10. Command 4 --> CS starts RS here as RS control flag gets reset to START at step-7
11. MI_BATCH_BUFFER_END

Workaround

Workaround:
Due to known HW issue "Resource Streamer Control" status of MI_RS_CONTROL command is not context save/restored across context switches. SW must ensure all pool allocations (3DSTATE_BINDING_TABLE_POOL_ALLOC, 3DSTATE_GATHER_POOL_ALLOC, 3DSTATE_DX9_CONSTANT_BUFFER_POOL_ALLOC) are disabled and no Resource Streamer specific commands are programmed when the "Resource Streamer Control" is programmed to "Stop".

MI_RS_CONTROL

Example:

....

MI_RS_CONTROL (Stop Resource Streamer)

3DSTATE_BINDING_TABLE_POOL_ALLOC (Binding Table Pool Disable)

3DSTATE_GATHER_POOL_ALLOC (Gather Pool Disable)

3DSTATE_DX9_CONSTANT_BUFFER_POOL_ALLOC (Constant Buffer Pool Disable)

//Following Commands must not be programmed

//3DSTATE_BINDING_TABLE_EDIT_*

//3DSTATE_GATHER_CONSTANT_*

//3DSTATE_DX9_CONSTANTF_*

....

MI_RS_CONTROL (Start Resource Streamer)

3DSTATE_BINDING_TABLE_POOL_ALLOC (Binding Table Pool Enable)

3DSTATE_GATHER_POOL_ALLOC (Gather Pool Enable)

3DSTATE_DX9_CONSTANT_BUFFER_POOL_ALLOC (Constant Buffer Pool Enable)

....

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 06h MI_RS_CONTROL Format: OpCode
	22:1	**Reserved** Format: MBZ
	0	**Resource Streamer Control** Format: U1 This bit specifies whether the command is starting or stopping the Resource Streamer.

Value	Name	Description
0h	Stop	Stop and disable the Resource Streamer
1h	Start	Start and enable the Resource Streamer

MI_RS_STORE_DATA_IMM

MI_RS_STORE_DATA_IMM

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The MI_RS_STORE_DATA_IMM command requests a write of the DWord constant supplied in the packet to the specified Memory Address.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 2Bh Format: OpCode MI_RS_STORE_DATA_IMM
	22	**Reserved** Project: CHV, BSW Format: MBZ
	21	**Reserved**
	20:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 2h Format: =n Total Length - 2. Excludes DWord (0,1).
1..2 **Project:** CHV, BSW	63:2	**Destination Address** Project: CHV, BSW Format: GraphicsAddress[63:2] This field specifies Bits 47:2 of the Address where the DWord will be stored. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47]. When render engine is PPGTT enabled this Address is translated using PPGTT, else GGTT is used for translation.
	1	**Reserved** Project: CHV, BSW Format: MBZ

MI_RS_STORE_DATA_IMM		
	0	**Core Mode Enable** Project: CHV, BSW If this bit is set then the address will be offset by the Core ID: If Core ID 0, then there is no offset If Core ID 1, then the Memory is offset by the size of the data.
3	31:0	**Data DWord 0** Format: U32 This field specifies the DWord value to be written to the targeted location.

MI_SEMAPHORE_SIGNAL

MI_SEMAPHORE_SIGNAL

Project:	CHV, BSW
Source:	CommandStreamer
Length Bias:	2

This command is used to signal the target engine stating the memory semaphore update occurrence to one of its contexts with **Target Context ID.** MI_SEMPHORE_SIGNAL and MI_SEMAPHORE_WAIT together replace the MI_SEMAPHORE_MBOX command. MI_ATOMIC (non-posted) command will be programmed prior to this command to update the semaphore data in memory.

Programming Notes	Source
[All Command Streamers]: When a semaphore signal is received by a target command streamer while context switch is in progress due to semaphore wait unsuccessful in signal mode, and the received semaphore signal is for the context getting switched out, Command Streamer might not forward the semaphore signal to GUC. As a result GUC might see a context with a switch reason as Semaphore Wait, for which it may never receive any semaphore signal; hence GUC might not schedule the same context forever. Since this issue is only applicable when MI_SEMAPHORE_WAIT is used in signal mode, SW has to WA this issue by doing one of the below: **SW Work Around:** 1. Scheduler on encountering a Context Waiting for semaphore signal to occur for a long time can assume above scenario could have occurred and do one of the below: a. Evaluate the semaphore wait condition based on the contexts PPHWSP semaphore wait details and re-schedule it, if the semaphore wait condition is satisfied. b. Schedule the context to HW and let HW evaluate the condition and take appropriate action. **OR** 2. Scheduler not to use MI_SEMAPHORE_WAIT in signal mode. Option 1 is preferred so that limited validation can be done for MI_SEMAPHORE_WAIT in signal mode on stepping's on which this issue is not fixed. **Example describing the scenarios causing issue:** RCS is executing Context-A. RCS has parsed MI_SEMAPHORE_WAIT in signal mode and has made memory request to fetch the semaphore data.	CommandStreamer

MI_SEMAPHORE_SIGNAL

BCS in the meantime update semaphore memory location for Context-A. BCS generates Semaphore Signal with Context ID as Context-A to RCS. RCS receives semaphore signal from BCS for Context-A. RCS receives the memory data and semaphore wait is un-successful (RCS must have sampled memory before BCS has updated the memory) resulting in context switch due to Wait on Semaphore. RCS ignores the semaphore signal received from BCS and also doesn't forward it to GUC. RCS switches out context-A with Wait on Semaphore as context switch reason. GUC process the context switch reason for Context-A, waits for semaphore signal for context-A to reschedule it which it will never receive as RCS has dropped it.	
[All Command Streamers]: When SW intends to use semaphore signaling between Command streamers, it must avoid lite restores in HW by programming "Force Restore" bit to ‘1’ in context descriptor during context submission, this is required to avoid known HW issue.	

Workaround	Project
Workaround: Post-Sync operation bit must not be set when Target Engine Select is set to RCS.	CHV, BSW

DWord	Bit	Description
0	31:29	**Command Type**
		Default Value: 0h MI_COMMAND
		Format: OpCode
	28:23	**MI Command Opcode**
		Default Value: 1Bh MI_SEMAPHORE_SIGNAL
		Format: OpCode
	22	**Reserved**
		Format: MBZ
	21	**Post-Sync Operation**
		Project: CHV, BSW
		Source: RenderCS

Value	Name	Description
0h	No Post Sync Operation	Command is executed as usual.
1h	Post Sync Operation	MI_SEMAPHORE_SIGNAL command is executed as a pipelined PIPE_CONTROL flush command with Semaphore Signal as post sync operation. Flush completion only guarantees the workload prior to this command is pushed till Windower unit and completion of any outstanding flushes issued prior to this command.

MI_SEMAPHORE_SIGNAL

		Programming Notes Any desired pipeline flush operation can be achieved by programming PIPE_CONTROL command prior to this command. When 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. When 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. This bit must not be set due to known HW issues. **Workaround** Workaround: "Post Sync Operation" bit must not be set when MI_SEMAPHPORE_SIGNAL command is programmed by GPGPU and MEDIA workloads (i.e when PIPELINE_SELECT command is set to GPGPU or MEDIA). This is to WA FFDOP CG issue, this WA need not be implemented when FF_DOP_CG is disabled via "Fixed Function DOP Clock Gate Disable" bit in RC_PSMI_CTRL register. Workaround: Post-Sync operation bit must not be set when Target Engine Select is set to RCS.
	21	**Reserved** Source: BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS Format: MBZ
	20:18	**Reserved** Format: MBZ
	17:15	**Target Engine Select** Project: CHV, BSW This field selects the target engine to which SIGNAL will be send to. Value | Name 0h | RCS 1h | VCS0 2h | BCS 3h | VECS 4h | VCS1 5h-7h | Reserved
	14:8	**Reserved** Format: MBZ
	7:0	**DWord Length**

MI_SEMAPHORE_SIGNAL

<table>
<tr><td></td><td></td><td><table><tr><td>Default Value:</td><td>0h</td></tr><tr><td>Format:</td><td>=n</td></tr></table>Total Length - 2. Excludes DWord (0,1).</td></tr>
<tr><td>1</td><td>31:0</td><td>Target Context ID<table><tr><th>Description</th></tr><tr><td>In execlist based scheduling this field contains the Context ID corresponding to the context of the target engine that this command is signaling. Target engine waiting on MI_SEMPHORE_WAIT in signal mode will re-fetch the data from memory or comparison if its context ID is same as this signaled Context ID. When execlists are enabled, Target engine on receiving this Context ID sends message to the SHIM if it doesn't have the context with the same Context ID running. Message send to SHIM carries the Context ID which will be looked at by UC for rescheduling the signaled Context ID. Target engine waiting on MI_SEMAPHORE_WAIT in signal mode will fetch data from memory for comparison on receiving signal irrespective of the context id received.</td></tr><tr><td>In ring buffer mode of scheduling this field doesn't have any relevance.</td></tr></table></td></tr>
</table>

MI_SEMAPHORE_WAIT

MI_SEMAPHORE_WAIT

Project:	CHV, BSW
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. This is for debug mode

MI_SEMPHORE_SIGNAL and MI_SEMAPHORE_WAIT together replace the MI_SEMAPHORE_MBOX command.

Programming Notes	Source
Render CS Only: SW must always program PIPE_CONTROL with "CS Stall" and "Render Target Cache Flush Enable" set prior to programming MI_SEMAPHORE_WAIT command for GPGPU workloads i.e when pipeline select is GPGPU via PIPELINE_SELECT command. This is required to achieve better GPGPU preemption latencies for certain programming sequences. If programming PIPE_CONTROL has performance implications then preemption latencies can be trade off against performance by not implementing this programming note.	RenderCS
Render CS Only: Ring Buffer Scheduling: CS doesn't generate semaphore wait interrupt to the scheduler when MI_SEMAPHORE_WAIT command is un-successful.	RenderCS
[All Command Streamers]: When a semaphore signal is received by a target command streamer while context switch is in progress due to semaphore wait unsuccessful in signal mode, and the received semaphore signal is for the context getting switched out, Command Streamer might not forward the semaphore signal to GUC. As a result GUC might see a context with a switch reason as Semaphore Wait, for which it may never receive any semaphore signal; hence GUC might not schedule the same context forever.	CommandStreamer

MI_SEMAPHORE_WAIT

Since this issue is only applicable when MI_SEMAPHORE_WAIT is used in signal mode, SW has to WA this issue by doing one of the below: **SW Work Around:** 1. Scheduler on encountering a Context Waiting for semaphore signal to occur for a long time can assume above scenario could have occurred and do one of the below: 1. Evaluate the semaphore wait condition based on the contexts PPHWSP semaphore wait details and re-schedule it, if the semaphore wait condition is satisfied. b. Schedule the context to HW and let HW evaluate the condition and take appropriate action. **OR** 2. Scheduler not to use MI_SEMAPHORE_WAIT in signal mode. Option 1 is preferred so that limited validation can be done for MI_SEMAPHORE_WAIT in signal mode on stepping's on which this issue is not fixed. **Example describing the scenarios causing issue:** RCS is executing Context-A. RCS has parsed MI_SEMAPHORE_WAIT in signal mode and has made memory request to fetch the semaphore data. BCS in the meantime update semaphore memory location for Context-A. BCS generates Semaphore Signal with Context ID as Context-A to RCS. RCS receives semaphore signal from BCS for Context-A. RCS receives the memory data and semaphore wait is un-successful (RCS must have sampled memory before BCS has updated the memory) resulting in context switch due to Wait on Semaphore. RCS ignores the semaphore signal received from BCS and also doesn't forward it to GUC. RCS switches out context-A with Wait on Semaphore as context switch reason. GUC process the context switch reason for Context-A, waits for semaphore signal for context-A to reschedule it which it will never receive as RCS has dropped it.	
[Ring Buffer Mode Of scheduling] [BlitterCS, VideoCS, VideoEnhancementCS, VideoCS2: Command Streamers Only]: HW loses Page Directory (PPGTT) information on becoming IDLE. SW must always program the PD information following MI_SEMAPHORE_WAIT command. This will ensure Page Directory information gets reprogrammed after exiting IDLE flow triggered on MI_SEMAPHORE_WAIT command. Alternatively SW can disable IDLE flows on MI_SEMAPHORE_WAIT by setting "Semaphore Wait Event IDLE Message Disable" bit in "BCS_ECOSKPD" register.	BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS
[VideoCS, VideoEnhancementCS, VideoCS2: Command Streamers Only]: MI_SEMAPHORE_WAIT cannot be executed in an Encrypted batch buffer(MI_BATCH_BUFFER_START, **Encrypt Enable**) while in Ring Buffer Mode (GFX_MODE bit 15).	VideoCS, VideoCS2, VideoEnhancementCS

Workaround
Workaround: [All Command Streamers][Ring Buffer Mode of Scheduling]: MI_SEMAPHORE_WAIT command must be always programmed with "Wait Mode" set to "Polling Mode" Or MI_SEMAPHORE_WAIT command with "Wait Mode" set to "Polling Mode" can be programmed when "Semaphore Wait Event IDLE message Disable" bit in "RC_PSMI_CTRL" register is set to disable Idle messaging on unsuccessful MI_SEMPAHORE_WAIT.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td>Command Type<table><tr><td>Default Value:</td><td>0h MI_COMMAND</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:23</td><td>MI Command Opcode<table><tr><td>Default Value:</td><td>1Ch MI_SEMAPHORE_WAIT</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>22</td><td>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.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Per Process Graphics Address</td><td></td></tr><tr><td>1h</td><td>Global Graphics Address</td><td>This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.</td></tr></table></td></tr>
<tr><td></td><td>21:18</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>17</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>16</td><td>Reserved<table><tr><td>Project:</td><td>CHV, BSW</td></tr></table></td></tr>
<tr><td></td><td>15</td><td>Wait Mode
This bit specifies the WAIT behavior when the semaphore comparison fails and before the context is switched out.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>1h</td><td>Polling Mode</td><td>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.</td></tr><tr><td>0h</td><td>Signal Mode</td><td>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.</td></tr></table></td></tr>
<tr><td></td><td>14:12</td><td>Compare Operation<table><tr><td>This field specifies the operation that will be executed to create the result that will either allow the context to continue or wait.</td></tr><tr><td>SAD = Semaphore Address Data
SDD = Semaphore Data Dword</td></tr></table></td></tr>
</table>

MI_SEMAPHORE_WAIT

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**

Default Value:	2h
Format:	=n Total Length - 2. Excludes DWord (0,1)

DWord 1, 31:0 **Semaphore Data Dword**

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.

DWord 2..3 **Project:** CHV, BSW

63:2 **Semaphore Address**

Project:	CHV, BSW
Format:	GraphicsAddress[63:2]

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**

Project:	CHV, BSW
Format:	MBZ

MI_SEMAPHORE_WAIT

MI_SEMAPHORE_WAIT

Project:	CHV, BSW
Source:	BlitterCS
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, same engine 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 BCS_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.
- BCS always generates an interrupt to the scheduler on encountering semaphore failure.

MI_SEMPHORE_SIGNAL and MI_SEMAPHORE_WAIT together replace the MI_SEMAPHORE_MBOX command.

Programming Notes

[Ring Buffer Mode Of scheduling][Video CS, Video Enhancement CS, Blitter CS]: HW loses Page Directory (PPGTT) information on becoming IDLE. SW must always program the PD information following MI_SEMAPHORE_WAIT command. This will ensure Page Directory information gets reprogrammed after exiting IDLE flow triggered on MI_SEMAPHORE_WAIT command. Alternatively SW can disable IDLE flows on MI_SEMAPHORE_WAIT by setting “Semaphore Wait Event IDLE Message Disable“ bit in “BCS_ECOSKPD” register.

When a semaphore signal is received by a target command streamer while context switch is in progress due to semaphore wait unsuccessful in signal mode, and the received semaphore signal is for the context getting switched out, Command Streamer might not forward the semaphore signal to GUC. As a result GUC might see a context with a switch reason as Semaphore Wait, for which it may never receive any semaphore signal; hence GUC might not schedule the same context forever. Since this issue is only applicable when MI_SEMAPHORE_WAIT is used in signal mode, SW has to WA this issue by doing one of the below: **SW Work Around:**

1. Scheduler on encountering a Context Waiting for semaphore signal to occur for a long time can assume above scenario could have occurred and do one of the below:

   a. Evaluate the semaphore wait condition based on the contexts PPHWSP semaphore wait details and re-schedule it, if the semaphore wait condition is satisfied. b. Schedule the context to HW and let HW evaluate the condition and take appropriate action. **OR**

MI_SEMAPHORE_WAIT

2. Scheduler not to use MI_SEMAPHORE_WAIT in signal mode.

Option 1 is preferred so that limited validation can be done for MI_SEMAPHORE_WAIT in signal mode on stepping's on which this issue is not fixed. **Example describing the scenarios causing issue:** RCS is executing Context-A. RCS has parsed MI_SEMAPHORE_WAIT in signal mode and has made memory request to fetch the semaphore data. BCS in the meantime update semaphore memory location for Context-A. BCS generates Semaphore Signal with Context ID as Context-A to RCS. RCS receives semaphore signal from BCS for Context-A. RCS receives the memory data and semaphore wait is un-successful (RCS must have sampled memory before BCS has updated the memory) resulting in context switch due to Wait on Semaphore. RCS ignores the semaphore signal received from BCS and also doesn't forward it to GUC. RCS switches out context-A with Wait on Semaphore as context switch reason. GUC process the context switch reason for Context-A, waits for semaphore signal for context-A to reschedule it which it will never receive as RCS has dropped it.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND 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.
	21:16	**Reserved**
	15	**Wait Mode** This bit specifies the WAIT behavior when the semaphore comparison fails and before the context is switched out. **Value** / **Name** / **Description** 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.

MI_SEMAPHORE_WAIT

DWord	Bit	Description
	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. If the below operation is TRUE then

SAD = Semaphore Address DataSDD = Semaphore Data Dword

Value	Name	Description
0h	SAD > SDD	If Indirect fetched data is greater than inline data then continue
1h	SAD >= SDD	If Indirect fetched data is greater than or equal to inline data then continue.
2h	SAD < SDD	If Indirect fetched data is less than inline data then continue.
3h	SAD <= SDD	If Indirect fetched data is less than or equal to inline data then continue.
4h	SAD == SDD	If Indirect fetched data is equalto inline data then continue.
5h	SAD != SDD	If Indirect fetched data is not equal to inline data then continue.
6h	Reserved	
7h	Reserved	

DWord	Bit	Description
	11:8	**Reserved**

Format:	MBZ

DWord	Bit	Description
	7:0	**DWord Length**

Default Value:	2h Excludes DWord (0,1)
Format:	=n

Total Length - 2

DWord	Bit	Description
1	31:0	**Semaphore Data Dword**

Format:	U32

Data dword to compare. The Data dword is supplied by software to control execution of the command buffer. If the data at Semaphore Address is greater than this dword, the execution of the command buffer continues.

DWord	Bit	Description
2..3	63:48	**Reserved**

Format:	MBZ

DWord	Bit	Description
	47:2	**Semaphore Address**

Format:	GraphicsAddress[47:2]

This field is the Graphics Memory Address of the 32 bit value for the semaphore.

DWord	Bit	Description
	1:0	**Reserved**

Format:	MBZ

MI_SEMAPHORE_WAIT

MI_SEMAPHORE_WAIT	
Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

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. This is for debug mode

MI_SEMPHORE_SIGNAL and MI_SEMAPHORE_WAIT together replace the MI_SEMAPHORE_MBOX command.

Programming Notes
Render CS Only: SW must always program PIPE_CONTROL with "CS Stall" and "Render Target Cache Flush Enable" set prior to programming MI_SEMAPHORE_WAIT command for GPGPU workloads i.e when pipeline select is GPGPU via PIPELINE_SELECT command. This is required to achieve better GPGPU preemption latencies for certain programming sequences. If programming PIPE_CONTROL has performance implications then preemption latencies can be trade off against performance by not implementing this programming note.
Render CS Only: Ring Buffer Scheduling: CS doesn't generate semaphore wait interrupt to the scheduler when MI_SEMAPHORE_WAIT command is un-successful.
When a semaphore signal is received by a target command streamer while context switch is in progress due to semaphore wait unsuccessful in signal mode, and the received semaphore signal is for the context getting switched out, Command Streamer might not forward the semaphore signal to GUC. As a result GUC might see a context with a switch reason as Semaphore Wait, for which it may never receive any semaphore signal; hence GUC might not schedule the same context forever. Since this issue is only applicable when MI_SEMAPHORE_WAIT is used in signal mode, SW has to WA this issue by doing one of the below: **SW Work Around:** 1. Scheduler on encountering a Context Waiting for semaphore signal to occur for a long time can

MI_SEMAPHORE_WAIT

assume above scenario could have occurred and do one of the below:

a. Evaluate the semaphore wait condition based on the contexts PPHWSP semaphore wait details and re-schedule it, if the semaphore wait condition is satisfied. b. Schedule the context to HW and let HW evaluate the condition and take appropriate action. **OR**

2. Scheduler not to use MI_SEMAPHORE_WAIT in signal mode.

Option 1 is preferred so that limited validation can be done for MI_SEMAPHORE_WAIT in signal mode on stepping's on which this issue is not fixed. **Example describing the scenarios causing issue:** RCS is executing Context-A. RCS has parsed MI_SEMAPHORE_WAIT in signal mode and has made memory request to fetch the semaphore data. BCS in the meantime update semaphore memory location for Context-A. BCS generates Semaphore Signal with Context ID as Context-A to RCS. RCS receives semaphore signal from BCS for Context-A. RCS receives the memory data and semaphore wait is un-successful (RCS must have sampled memory before BCS has updated the memory) resulting in context switch due to Wait on Semaphore. RCS ignores the semaphore signal received from BCS and also doesn't forward it to GUC. RCS switches out context-A with Wait on Semaphore as context switch reason. GUC process the context switch reason for Context-A, waits for semaphore signal for context-A to reschedule it which it will never receive as RCS has dropped it.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td>Command Type
<table><tr><td>Default Value:</td><td>0h MI_COMMAND</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:23</td><td>MI Command Opcode
<table><tr><td>Default Value:</td><td>1Ch MI_SEMAPHORE_WAIT</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>22</td><td>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.
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Per Process Graphics Address</td><td></td></tr>
<tr><td>1h</td><td>Global Graphics Address</td><td>This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.</td></tr></table></td></tr>
<tr><td></td><td>21:16</td><td>Reserved</td></tr>
<tr><td></td><td>15</td><td>Wait Mode
This bit specifies the WAIT behavior when the semaphore comparison fails and before the context is switched out.
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>1h</td><td>Polling Mode</td><td>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.</td></tr></table></td></tr>
</table>

MI_SEMAPHORE_WAIT

		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.
	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**
	7:0	**DWord Length** Default Value: 2h Format: =n Total Length - 2. Excludes DWord (0,1)
1	31:0	**Semaphore Data Dword** Format: U32 Data dword to compare. The Data dword is supplied by software to control execution of the command buffer. If the data at **Semaphore Address** is greater than this dword, the execution of the command buffer continues.
2	31:2	**Semaphore Address** Format: GraphicsAddress[31:2] This field is the Graphics Memory Address of the 32-bit value for the semaphore.
	1:0	**Reserved**
3	31:16	**Reserved**

MI_SEMAPHORE_WAIT		
	15:0	**Semaphore Address High** Format: GraphicsAddress[47:32] This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space.

MI_SEMAPHORE_WAIT

MI_SEMAPHORE_WAIT	
Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

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, same engine 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 VCS_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.
- VCS always generates an interrupt to the scheduler on encountering semaphore failure.

MI_SEMPHORE_SIGNAL and MI_SEMAPHORE_WAIT together replace the MI_SEMAPHORE_MBOX command.

Programming Notes
MI_SEMAPHORE_WAIT cannot be executed in an Encrypted batch buffer(MI_BATCH_BUFFER_START, **Encrypt Enable**) while in Ring Buffer Mode (GFX_MODE bit 15).
[Ring Buffer Mode Of scheduling][Video CS]: HW loses Page Directory (PPGTT) information on becoming IDLE. SW must always program the PD information following MI_SEMAPHORE_WAIT command. This will ensure Page Directory information gets reprogrammed after exiting IDLE flow triggered on MI_SEMAPHORE_WAIT command. Alternatively SW can disable IDLE flows on MI_SEMAPHORE_WAIT command by setting “Semaphore Wait Event IDLE Message Disable“ bit in “VCS_ECOSKPD” register.
When a semaphore signal is received by a target command streamer while context switch is in progress due to semaphore wait unsuccessful in signal mode, and the received semaphore signal is for the context getting switched out, Command Streamer might not forward the semaphore signal to GUC. As a result GUC might see a context with a switch reason as Semaphore Wait, for which it may never receive any semaphore signal; hence GUC might not schedule the same context forever. Since this issue is only applicable when MI_SEMAPHORE_WAIT is used in signal mode, SW has to WA this issue by doing one of the below: **SW Work Around:** 1. Scheduler on encountering a Context Waiting for semaphore signal to occur for a long time can assume above scenario could have occurred and do one of the below: a. Evaluate the semaphore wait condition based on the contexts PPHWSP semaphore wait details and re-schedule it, if the semaphore wait condition is satisfied. b. Schedule the context to HW and let HW evaluate the condition and take appropriate action. **OR**

MI_SEMAPHORE_WAIT

2. Scheduler not to use MI_SEMAPHORE_WAIT in signal mode.

Option 1 is preferred so that limited validation can be done for MI_SEMAPHORE_WAIT in signal mode on stepping's on which this issue is not fixed. **Example describing the scenarios causing issue:** RCS is executing Context-A. RCS has parsed MI_SEMAPHORE_WAIT in signal mode and has made memory request to fetch the semaphore data. BCS in the meantime update semaphore memory location for Context-A. BCS generates Semaphore Signal with Context ID as Context-A to RCS. RCS receives semaphore signal from BCS for Context-A. RCS receives the memory data and semaphore wait is un-successful (RCS must have sampled memory before BCS has updated the memory) resulting in context switch due to Wait on Semaphore. RCS ignores the semaphore signal received from BCS and also doesn't forward it to GUC. RCS switches out context-A with Wait on Semaphore as context switch reason. GUC process the context switch reason for Context-A, waits for semaphore signal for context-A to reschedule it which it will never receive as RCS has dropped it.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND 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: 0h: Per Process Graphics Address
	21:16	**Reserved** Format: MBZ
	15	**Wait Mode** This bit specifies the WAIT behavior when the semaphore comparison fails and before the context is switched out. Value / Name / Description: 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.
	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. If the below operation is TRUE then

MI_SEMAPHORE_WAIT

Value	Name	Description
0h	SAD > SDD	If Indirect fetched data is greater than inline data then continue.
1h	SAD >= SDD	If Indirect fetched data is greater than or equal to inline data then continue.
2h	SAD < SDD	If Indirect fetched data is less than inline data then continue.
3h	SAD <= SDD	If Indirect fetched data is less than or equal to inline data then continue.
4h	SAD == SDD	If Indirect fetched data is equalto inline data then continue.
5h	SAD != SDD	If Indirect fetched data is not equal to inline data then continue.
6h	Reserved	
7h	Reserved	

Programming Notes
SAD = Semaphore Address DataSDD = Semaphore Data Dword

11:8 Reserved

Format:	MBZ

7:0 DWord Length

Default Value:	2h Excludes DWord (0,1)
Format:	=n

Total Length - 2

DWord 1

31:0 Semaphore Data Dword

Format:	U32

Data dword to compare. The Data dword is supplied by software to control execution of the command buffer. If the data at **Semaphore Address** is greater than this dword, the execution of the command buffer continues.

DWord 2

31:2 Semaphore Address

Format:	GraphicsVirtualAddress[31:2]

This field is the Graphics Memory Address of the 32 bit value for the semaphore.

1:0 Reserved

Format:	MBZ

DWord 3

31:16 Reserved

Format:	MBZ

15:0 Semaphore Address High

Format:	GraphicsAddress[47:32]

This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space.

MI_SEMAPHORE_WAIT

MI_SEMAPHORE_WAIT	
Project:	CHV, BSW
Source:	VideoEnhancementCS
Length Bias:	2

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, same engine 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 VECS_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.
- VECS always generates an interrupt to the scheduler on encountering semaphore failure.

MI_SEMPHORE_SIGNAL and MI_SEMAPHORE_WAIT together replace the MI_SEMAPHORE_MBOX command.

Programming Notes
[Ring Buffer Mode Of scheduling][Video Enhancement CS]: HW loses Page Directory (PPGTT) information on becoming IDLE. SW must always program the PD information following MI_SEMAPHORE_WAIT command. This will ensure Page Directory information gets reprogrammed after exiting IDLE flow triggered on MI_SEMAPHORE_WAIT command. Alternatively SW can disable IDLE flows on MI_SEMAPHORE_WAIT by setting “Semaphore Wait Event IDLE Message Disable“ bit in “VECS_ECOSKPD” register.
When a semaphore signal is received by a target command streamer while context switch is in progress due to semaphore wait unsuccessful in signal mode, and the received semaphore signal is for the context getting switched out, Command Streamer might not forward the semaphore signal to GUC. As a result GUC might see a context with a switch reason as Semaphore Wait, for which it may never receive any semaphore signal; hence GUC might not schedule the same context forever. Since this issue is only applicable when MI_SEMAPHORE_WAIT is used in signal mode, SW has to WA this issue by doing one of the below: **SW Work Around:** 1. Scheduler on encountering a Context Waiting for semaphore signal to occur for a long time can assume above scenario could have occurred and do one of the below: a. Evaluate the semaphore wait condition based on the contexts PPHWSP semaphore wait details and re-schedule it, if the semaphore wait condition is satisfied. b. Schedule the context to HW and let HW evaluate the condition and take appropriate action. **OR** 2. Scheduler not to use MI_SEMAPHORE_WAIT in signal mode. Option 1 is preferred so that limited validation can be done for MI_SEMAPHORE_WAIT in signal mode on

MI_SEMAPHORE_WAIT

stepping's on which this issue is not fixed. **Example describing the scenarios causing issue:** RCS is executing Context-A. RCS has parsed MI_SEMAPHORE_WAIT in signal mode and has made memory request to fetch the semaphore data. BCS in the meantime update semaphore memory location for Context-A. BCS generates Semaphore Signal with Context ID as Context-A to RCS. RCS receives semaphore signal from BCS for Context-A. RCS receives the memory data and semaphore wait is un-successful (RCS must have sampled memory before BCS has updated the memory) resulting in context switch due to Wait on Semaphore. RCS ignores the semaphore signal received from BCS and also doesn't forward it to GUC. RCS switches out context-A with Wait on Semaphore as context switch reason. GUC process the context switch reason for Context-A, waits for semaphore signal for context-A to reschedule it which it will never receive as RCS has dropped it.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND 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: 0h — Name: Per Process Graphics Address
	21:16	**Reserved** Format: MBZ
	15	**Wait Mode** This bit specifies the WAIT behavior when the semaphore comparison fails and before the context is switched out. Value 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. Value 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.
	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. If the below operation is TRUE then

MI_SEMAPHORE_WAIT

		<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>SAD > SDD</td><td>If Indirect fetched data is greater than inline data then continue.</td></tr><tr><td>1h</td><td>SAD >= SDD</td><td>If Indirect fetched data is greater than or equal to inline data then continue.</td></tr><tr><td>2h</td><td>SAD < SDD</td><td>If Indirect fetched data is less than inline data then continue.</td></tr><tr><td>3h</td><td>SAD <= SDD</td><td>If Indirect fetched data is less than or equal to inline data then continue.</td></tr><tr><td>4h</td><td>SAD == SDD</td><td>If Indirect fetched data is equalto inline data then continue.</td></tr><tr><td>5h</td><td>SAD != SDD</td><td>If Indirect fetched data is not equal to inline data then continue.</td></tr><tr><td>6h</td><td>Reserved</td><td></td></tr><tr><td>7h</td><td>Reserved</td><td></td></tr></table> <table><tr><th>Programming Notes</th></tr><tr><td>SAD = Semaphore Address DataSDD = Semaphore Data Dword</td></tr></table>
	11:8	**Reserved**
	7:0	**DWord Length** <table><tr><td>Default Value:</td><td>2h Excludes DWord (0,1)</td></tr><tr><td>Format:</td><td>=n</td></tr></table>Total Length - 2
1	31:0	**Semaphore Data Dword** <table><tr><td>Format:</td><td>U32</td></tr></table>Data dword to compare. The Data dword is supplied by software to control execution of the command buffer. If the data at **Semaphore Address** is greater than this dword, the execution of the command buffer continues.
2	31:2	**Semaphore Address** <table><tr><td>Format:</td><td>GraphicsVirtualAddress[31:2]</td></tr></table>This field is the Graphics Memory Address of the 32 bit value for the semaphore.
	1:0	**Reserved**
3	31:16	**Reserved** <table><tr><td>Format:</td><td>MBZ</td></tr></table>
	15:0	**Semaphore Address High** <table><tr><td>Format:</td><td>GraphicsAddress[47:32]</td></tr></table>This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space.

MI_SET_CONTEXT

MI_SET_CONTEXT

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The MI_SET_CONTEXT command is used to specify the logical context associated with the hardware context. A logical context is an area in memory used to store hardware context information, and the context is referenced via a 2KB-aligned pointer. If the (new) logical context is different (i.e., at a different memory address), the device saves the current HW context values to the current logical context address, and then restores (loads) the new logical context by reading the context from the new address and loading it into the hardware context state. If the logical context address specified in this command matches the current logical context address, this command is effectively treated as a NOOP. Specific to the Render command stream only. This command also includes some controls over the context save/restore process. The Force Restore bit can be used to refresh the on-chip device state from the same memory address if the indirect state buffers have been modified. The Restore Inhibit bit can be used to prevent the new context from being loaded at all. This must be used to prevent an uninitialized context from being loaded. Once software has initialized a context (by setting all state variables to initial values via commands), the context can then be stored and restored normally. When switching from a generic media context to a 3D context, the generic media state must be cleared via the Generic Media State Clear bit 16 in PIPE_CONTROL (or bit 4 in MI_FLUSH) before saving 3D context. MI_SET_CONTEXT commands are permitted only within a ring buffer (not within a batch buffer).

Programming Notes
This command is legal only if Execlist Enable in the GFX_MODE register is reset. Otherwise, execlists must be used to switch context in lieu of MI_SET_CONTEXT.
This command needs to be always followed by a single MI_NOOP instruction to workaround a silicon issue.
[CHV, BSW]: MI_ARB_ON_OFF with 'Arbitration Enable Reset' set should be programmed before an MI_SET_CONTEXT command. MI_ARB_ON_OFF with 'Arbitration Enable' set should be programmed after an MI_SET_CONTEXT command. This programming ensures that PSMI context switch flows do not conflict with MI_SET_CONTEXT flows.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 18h MI_SET_CONTEXT Format: OpCode
	22:8	**Reserved**
	7:0	**DWord Length** Default Value: 0h Format: =n Total Length - 2. Excludes DWord (0,1).

MI_SET_CONTEXT

DWord	Bit	Description
1	31:12	**Logical Context Address** Project: CHV, BSW Format: GraphicsAddress[31:12]LogicalContext **Description** This field contains the 4KB-aligned graphics memory address of the Logical Context that is to be loaded into the hardware context. If this address is equal to the CCID register associated with the current ring, no load will occur. Prior to loading this new context, the device will save the existing context as required. After the context switch operation completes, this address will be loaded into the associated CCID register. This field needs to be 4KB aligned virtual address.
	11:10	**Reserved** Format: MBZ
	9	**Reserved**
	8	**Reserved, Must be 1** Format: Must Be One
	7:5	**Reserved**
	4	**Core Mode Enable** Project: CHV, BSW Format: Enable If set the Context Image will be offset based off the Core ID: If Core ID 0, no offset If Core ID 1, 36KB Offset
	3	**Resource Streamer State Save Enable** Project: CHV, BSW Format: Enable If set, the resource streamer state identified in the Logical Context Data section of the Memory Data Formats chapter is saved as part of switching away from this logical context. This bit will be stored in the associated CCID register to control the context save operation when switching away from this context (as part of a subsequent MI_SET_CONTEXT command).
	2	**Resource Streamer State Restore Enable** Project: CHV, BSW Format: Enable If set, the resource streamer state identified in the Logical Context Data section of the Memory Data Formats chapter is loaded (or restored) as part of switching to this logical context. This bit affects the switch (if required) to the context specified in Logical Context Address. This bit will also be stored in the associated CCID register to control a subsequent context save operation when switching to this context (as part of a subsequent ring buffer switch).

MI_SET_CONTEXT		
	1	**Force Restore** When switching to this logical context a comparison between Logical Context Address and the contests of the CCID register is performed. Normally, matching addresses prevent a context restore from occurring; however, when this bit is set a context restore is forced to occur. This bit cannot be set with Restore Inhibit. Note: This bit is not saved in the associated CCID register. It only affects the processing of this command.
	0	**Restore Inhibit** If set, the restore of the HW context from the logical context specified by Logical Context Address is inhibited (i.e., the existing HW context values are maintained). This bit must be used to prevent the loading of an uninitialized logical context. If clear, the context switch proceeds normally. This bit cannot be set with Force Restore. Note: This bit is not saved in the associated CCID register. It only affects the processing of this command.

MI_SET_PREDICATE

MI_SET_PREDICATE

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	1

Description

This command sets the Predication Check for the subsequent commands in the command buffer except for MI_SET_PREDICATE itself. Render Command Streamer NOOPs the following commands based on the PREDICATE_ENABLE from MI_SET_PREDICATE, MI_SET_PREDICATE_RESULT and MI_SET_PREDICATE_RESULT_2 status. Resource Streamer doesn't take any action of parsing MI_SET_PREDICATE, this command is similar to any other command which is not meant for resource streamer.

Executing MI_SET_PREDICATE command sets PREDICATE_ENABLE bits in MI_MODE register, MI_MODE register gets render context save restored.

Programming Notes

- MI_SET_PREDICATE predication scope must be confined within a Batch Buffer to set of commands.
- MI_SET_PREDICATE with Predicate Enable Must always have a corresponding MI_SET_PREDICATE with Predicate Disable within the same Batch Buffer.
- MI_ARB_CHK command must be programmed outside the Predication Scope of MI_SET_PREDICATE.
- MI_SET_PREDICATE Predication Scope must not involve any RC6 triggering events.

Only the following command(s) can be programmed between the MI_SET_PREDICATE command enabled for predication: 3DSTATE_URB_VS 3DSTATE_URB_HS 3DSTATE_URB_DS 3DSTATE_URB_GS 3DSTATE_PUSH_CONSTANT_ALLOC_VS 3DSTATE_PUSH_CONSTANT_ALLOC_HS 3DSTATE_PUSH_CONSTANT_ALLOC_DS 3DSTATE_PUSH_CONSTANT_ALLOC_GS 3DSTATE_PUSH_CONSTANT_ALLOC_PS MI_LOAD_REGISTER_IMM MEDIA_VFE_STATE MEDIA_OBJECT MEDIA_OBJJECT_WALKER MEDIA_INTERFACE_DESCRIPTOR_LOAD 3DSTATE_WM_HZ_OP

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 01h MI_SET_PREDICATE Format: OpCode
	22:4	**Reserved**
	3:0	**PREDICATE ENABLE** Project: CHV, BSW This field sets the predication logic in render command streamer when parsed. Predicate Disable is the default mode of operation.

MI_SET_PREDICATE

Value	Name	Description
0h	NOOP Never	Predication is Disabled and RCS will process commands as usual.
1h	NOOP on Result2 clear	Following Commands will be NOOPED by RCS only if the MI_PREDICATE_RESULT_2 is clear.
2h	NOOP on Result2 set	Following Commands will be NOOPED by RCS only if the MI_PREDICATE_RESULT_2 is set.
3h	NOOP on Result clear	Following Commands will be NOOPED by RCS only if the MI_PREDICATE_RESULT is clear.
4h	NOOP on Result set	Following Commands will be NOOPED by RCS only if the MI_PREDICATE_RESULT is set.
5h	Execute when one slice enabled.	Following Commands will be Executed by RCS only when one slice is enabled.
6h	Execute when two slices are enabled.	Following Commands will be Executed by RCS only when two slices are enabled.
7h	Execute when three slices are enabled.	Following Commands will be Executed by RCS only when all the three slices are enabled.
8h-Ah	Reserved	
Bh, Ch	Reserved	
Dh, Eh	Reserved	
Fh	NOOP Always	Following Commands will be NOOPED by RCS unconditionally.

MI_STORE_DATA_IMM

MI_STORE_DATA_IMM

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

The MI_STORE_DATA_IMM command requests a write of the QWord or DWord constant supplied in the packet to the specified Memory Address. This command also supports writing to consecutive dword or qword memory locations form the starting address. As the write targets a System Memory Address, the write operation is coherent with the CPU cache (i.e., the processor cache is snooped).

Programming Notes
This command can be used for general software synchronization through variables in cacheable memory (i.e., where software does not need to poll un-cached memory or device registers).
This command simply initiates the write operation with command execution proceeding normally. Although the write operation is guaranteed to complete "eventually", there is no mechanism to synchronize command execution with the completion (or even initiation) of these operations.
This command should not be used within a non_privilege batch buffer to access global virtual space, doing so will be treated as privilege access violation. Refer "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 20h MI_STORE_DATA_IMM Format: OpCode
	22	**Use Global GTT** Format: U1 If set, this command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch or ring buffer. If clear, the PPGTT will be used. 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.
	21	**Store Qword** Format: U1

Value	Name	Description
0h	Store Dword	If set, this command generates dword writes to memory. Number of dwords generated depends upon the number of 'Data Dword' programmed in the command. If 'x' number of data dwords are programmed in the command it results in 'x' dword writes to memory.

MI_STORE_DATA_IMM

DWord	Bit	Description
		1h | Store Qword | If set, this command generates Qword writes to memory, two 'Data Dword' are paired to form a Qword. Number of qwords generated depends upon the number of 'Data Dword' programmed in the command. If 'x' number of data dwords are programmed in the command it results in 'x/2' qword writes to memory.
	20:10	**Reserved** Format: MBZ
	9:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1..2	63:2	**Destination Address** Format: GraphicsAddress[63:2] This field specifies the 4GB aligned base address within the host's 64-bit virtual address space. As the store address must be DWord-aligned, Bits 1:0 of that address MBZ. This address must be 8B aligned if "Store Qword" is enabled. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].
	1:0	**Reserved** Format: MBZ
3	31:0	**Data DWord 0** Format: U32 FormatDesc This field specifies the DWord value to be written to the targeted location.For a QWord write this DWord is the lower DWord of the QWord to be reported (DW 0).
4	31:0	**Data DWord 1** Format: U32 FormatDesc This field specifies the upper DWord value to be written to the targeted QWord location (DW 1).

MI_STORE_DATA_IMM

MI_STORE_DATA_IMM

Project:	CHV, BSW
Source:	VideoEnhancementCS
Length Bias:	2

The MI_STORE_DATA_IMM command requests a write of the QWord or DWord constant supplied in the packet to the specified Memory Address. This command also supports writing to consecutive dword or qword memory locations form the starting address. As the write targets a System Memory Address, the write operation is coherent with the CPU cache (i.e., the processor cache is snooped).

Programming Notes
This command should not be used within a "non_privilege" batch buffer to access global virtual space, doing so will be treated as privilege access violation. Refer "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation. This command can be used within ring buffers and/or "privilege" batch buffers. If used within a non-privilege batch buffer, **Use Global GTT** must be clear. This command can be used for general software synchronization through variables in cacheable memory (i.e., where software does not need to poll un-cached memory or device registers). This command simply initiates the write operation with command execution proceeding normally. Although the write operation is guaranteed to complete "eventually", there is no mechanism to synchronize command execution with the completion (or even initiation) of these operations.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="4">0</td><td>31:29</td><td>Command Type
<table><tr><td>Default Value:</td><td>0h MI_COMMAND</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td>28:23</td><td>MI Command Opcode
<table><tr><td>Default Value:</td><td>20h MI_STORE_DATA_IMM</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td>22</td><td>Use Global GTT
<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>U1</td></tr></table>
If set, this command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer. If clear, the PPGTT will be used. 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.</td></tr>
<tr><td>21</td><td>Store Qword
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Store Dword</td><td>If set, this command generates dword writes to memory. Number of dwords generated depends upon the number of 'Data Dword' programmed in the command. If 'x' number of data dwords are programmed in the command it results in 'x' dword writes to memory.</td></tr></table></td></tr>
</table>

MI_STORE_DATA_IMM

DWord	Bit	Description
		1h — Store Qword — If set, this command generates Qword writes to memory, two 'Data Dword' are paired to form a Qword. Number of qwords generated depends upon the number of 'Data Dword' programmed in the command. If 'x' number of data dwords are programmed in the command it results in 'x/2' qword writes to memory.
	20:10	**Reserved** Project: All Format: MBZ
	9:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:2	**Address** Format: GraphicsAddress[31:2]U32(2) This field specifies Bits 31:2 of the Address where the DWord will be stored. As the store address must be DWord-aligned, Bits 1:0 of that address MBZ. This address must be 8B aligned for a store "QW" command.
	1:0	**Reserved** Format: MBZ
2	31:16	**Reserved** Format: MBZ
	15:0	**Address High** Format: GraphicsAddress[47:32]U16 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	**Data DWord 0** Format: U32 This field specifies the DWord value to be written to the targeted location.For a QWord write this DWord is the lower DWord of the QWord to be reported (DW 0).
4	31:0	**Data DWord 1** Format: U32 This field specifies the upper DWord value to be written to the targeted QWord location (DW 1).

MI_STORE_DATA_IMM

MI_STORE_DATA_IMM

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

Description
The MI_STORE_DATA_IMM command requests a write of the QWord constant supplied in the packet to the specified Memory Address. As the write targets a System Memory Address, the write operation is coherent with the CPU cache (i.e., the processor cache is snooped).
This command supports writing to multiple consecutive dword or qword memory locations from the starting address.

Programming Notes
This command can be used for general software synchronization through variables in cacheable memory (i.e., where software does not need to poll un-cached memory or device registers). However, the cacheable nature of the transaction is determined by the setting of the "mapping type" in the GTT entry. This command simply initiates the write operation with command execution proceeding normally. Although the write operation is guaranteed to complete "eventually", there is no mechanism to synchronize command execution with the completion (or even initiation) of these operations. All writes to memory generated using this command are expected to finish in order.
This command should not be used within a "non_privilege" batch buffer to access global virtual space, doing so will be treated as privilege access violation. Refer "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND
	28:23	**MI Command Opcode** Default Value: 20h MI_STORE_DATA_IMM
	22	**Use Global GTT** Project: All 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.

<table>
<tr><th colspan="3">MI_STORE_DATA_IMM</th></tr>
<tr><td rowspan="3"></td><td>21</td><td>Store Qword<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Store Dword</td><td>If set, this command generates dword writes to memory. Number of dwords generated depends upon the number of "Data Dword" programmed in the command. If 'x' number of data dwords are programmed in the command it results in "x" dword writes to memory.</td></tr><tr><td>1h</td><td>Store Qword</td><td>If set, this command generates Qword writes to memory, two "Data Dword" are paired to form a Qword. Number of qwords generated depends upon the number of "Data Dword" programmed in the command. If 'x' number of data dwords are programmed in the command it results in "x/2" qword writes to memory.</td></tr></table></td></tr>
<tr><td>20:10</td><td>Reserved</td></tr>
<tr><td>9:0</td><td>DWord Length<table><tr><td>Default Value:</td><td>2h Excludes DWord (0,1) = 2 for DWord, 3 for QWord</td></tr><tr><td>Format:</td><td>=n Total Length - 2</td></tr></table></td></tr>
<tr><td rowspan="3">1
Project:
CHV, BSW</td><td>31:2</td><td>Address<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>GraphicsAddress[31:2]U32(2)</td></tr></table>This field specifies Bits 31:2 of the Address where the DWord will be stored. As the store address must be DWord-aligned, Bits 1:0 of that address MBZ. This address must be 8B aligned for a store "QW" command.</td></tr>
<tr><td>1</td><td>Reserved</td></tr>
<tr><td>0</td><td>Core Mode Enable<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>U1</td></tr></table>This bit is set then the address will be offset by the Core ID: If Core ID 0, then there is no offset If Core ID 1, then the Memory is offset by the size of the data(32b or 64b based off number of DW length).</td></tr>
<tr><td rowspan="2">2
Project:
CHV, BSW</td><td>31:16</td><td>Reserved</td></tr>
<tr><td>15:0</td><td>Address High<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>GraphicsAddress[47:32]U32(2)</td></tr></table>This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space.</td></tr>
<tr><td>3</td><td>31:0</td><td>Data DWord 0<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>U32</td></tr></table>This field specifies the DWord value to be written to the targeted location.For a QWord write this DWord is the lower DWord of the QWord to be reported (DW 0).</td></tr>
</table>

<table>
<tr><th colspan="3">MI_STORE_DATA_IMM</th></tr>
<tr><td>4</td><td>31:0</td><td>Data DWord 1<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>U32</td></tr></table>This field specifies the upper DWord value to be written to the targeted QWord location (DW 1).</td></tr>
</table>

MI_STORE_DATA_IMM

MI_STORE_DATA_IMM

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

Description
The MI_STORE_DATA_IMM command requests a write of the QWord constant supplied in the packet to the specified Memory Address. As the write targets a System Memory Address, the write operation is coherent with the CPU cache (i.e., the processor cache is snooped).
This command supports writing to multiple consecutive dwords or qwords memory locations from the starting address.

Programming Notes

- This command should not be used within a "non-privilege" batch buffer to access global virtual space, doing so will be treated as privilege access violation. Refer "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation. This command can be used within ring buffers and/or privilege batch buffers to access global virtual space.
- This command can be used for general software synchronization through variables in cacheable memory (i.e., where software does not need to poll un-cached memory or device registers).
- This command simply initiates the write operation with command execution proceeding normally. Although the write operation is guaranteed to complete eventually, there is no mechanism to synchronize command execution with the completion (or even initiation) of these operations.

Number of consecutive dwords or qwords programmed must be restricted such that the DWord Length doesn't exceed 0x3FE, i.e single command supports updating 1021 consecutive dword locations or 510 qword locations.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 20h MI_STORE_DATA_IMM Format: OpCode
	22	**Use Global GTT** Project: All Format: Boolean

<table>
<tr><th colspan="3">MI_STORE_DATA_IMM</th></tr>
<tr><td></td><td></td><td><table><tr><th>Description</th></tr><tr><td>If set, this command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer. If clear, the PPGTT will be used. 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.</td></tr></table></td></tr>
<tr><td></td><td>21</td><td>Store Qword<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>Boolean</td></tr></table>If set, this command generates Qword writes to memory, two "Data Dword" are paired to form a Qword. Number of qwords generated depends upon the number of "Data Dword" programmed in the command. If 'x' number of "Data Dwords" are programmed in this command it results in "x/2" qword writes to memory. If reset this command generates Dwords writes to memory. Number of dwords generated depends upon the number of "Data Dword" programmed in the command. If 'x' number of "Data Dwords" are programmed in this command it results in "x" dword writes to memory.</td></tr>
<tr><td></td><td>20:10</td><td>Reserved</td></tr>
<tr><td></td><td>9:0</td><td>DWord Length<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>=n Total Length - 2. Excludes DWord (0,1)</td></tr></table><table><tr><th>Value</th><th>Name</th></tr><tr><td>2h</td><td>Store Dword [Default]</td></tr><tr><td>3h</td><td>Store Qword</td></tr></table><table><tr><th>Programming Notes</th></tr><tr><td>DWord Length programmed must not exceed 0x3FE.</td></tr><tr><td>If RS is enabled in the batch buffer, then the value of this field must not exceed 0x3F.</td></tr></table></td></tr>
<tr><td>1..2
Project:
CHV, BSW</td><td>63:48</td><td>Reserved</td></tr>
<tr><td></td><td>47:2</td><td>Address<table><tr><td>Format:</td><td>GraphicsAddress[47:2]</td></tr></table>This field specifies Bits 47:2 of the Address where the DWord will be stored. As the store address must be DWord-aligned, Bits 1:0 of that address MBZ. This address must be 8B aligned for a store "QW" command.</td></tr>
<tr><td></td><td>1</td><td>Reserved</td></tr>
<tr><td></td><td>0</td><td>Core Mode Enable<table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>U1</td></tr></table>This bit is set then the address will be offset by the Core ID: If Core ID 0, then there is no offset If Core ID 1, then the Memory is offset by the size of the data(32b or 64b based off number of DW length).</td></tr>
</table>

MI_STORE_DATA_IMM

3	31:0	**Data DWord 0** Format: U32 This field specifies the DWord value to be written to the targeted location.For a QWord write this DWord is the lower DWord of the QWord to be reported (DW 0).
4	31:0	**Data DWord 1** Format: U32 This field specifies the upper DWord value to be written to the targeted QWord location (DW 1).

MI_STORE_DATA_INDEX

MI_STORE_DATA_INDEX

Project:	CHV, BSW
Source:	VideoEnhancementCS
Length Bias:	2

The MI_STORE_DATA_INDEX command requests a write of the data constant supplied in the packet to the specified offset from the System Address defined by the Hardware Status Page Address Register. As the write targets a System Address, the write operation is coherent with the CPU cache (i.e., the processor cache is snooped).

Programming Notes

- Use of this command with an invalid or uninitialized value in the Hardware Status Page Address Register is UNDEFINED.
- This command can be used for general software synchronization through variables in cacheable memory (i.e., where software does not need to poll uncached memory or device registers).
- This command simply initiates the write operation with command execution proceeding normally. Although the write operation is guaranteed to complete "eventually", there is no mechanism to synchronize command execution with the completion (or even initiation) of these operations.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 21h MI_STORE_DATA_INDEX Format: OpCode
	22	**Reserved** Project: All Format: MBZ
	21	**Use Per-Process Hardware Status Page** Project: CHV, BSW If this bit is set, this command will index into the per-process hardware status page at offset 0K from the LRCA. If clear, the Global Hardware Status Page will be indexed. This bit must be '0' if the Execlist Enable bit is clear.
	20:8	**Reserved** Project: All Format: MBZ

MI_STORE_DATA_INDEX

DWord	Bit	Description
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) = 2 for QWord Project: All Format: =n Total Length - 2
1	31:12	**Reserved** Project: All Format: MBZ
	11:2	**Offset** Project: All Format: U10 Zero-based DWord offset into the HW status page Format: GraphicsAddress[11:2]U32 This field specifies the offset (into the hardware status page) to which the data will be written. Note that the first few DWords of this status page are reserved for special-purpose data storage - targeting these reserved locations via this command is UNDEFINED. For a QWord write, the offset is valid down to bit 3 only. Value: [16, 1023] — Name: (blank)
	1:0	**Reserved** Project: All Format: MBZ
2	31:0	**Data DWord 0** Format: U32 This field specifies the upper DWord value to be written to the targeted QWord location (DW 1).
3	31:0	**Data Word 1** Format: U32 This field specifies the upper DWord value to be written to the targeted QWord location (DW 1).

MI_STORE_DATA_INDEX

MI_STORE_DATA_INDEX

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

The MI_STORE_DATA_INDEX command requests a write of the data constant supplied in the packet to the specified offset from the System Address defined by the Hardware Status Page Address Register. As the write targets a System Address, the write operation is coherent with the CPU cache (i.e., the processor cache is snooped).

Programming Notes

Use of this command with an invalid or uninitialized value in the Hardware Status Page Address Register is UNDEFINED. This command can be used for general software synchronization through variables in cacheable memory (i.e., where software does not need to poll uncached memory or device registers). This command simply initiates the write operation with command execution proceeding normally. Although the write operation is guaranteed to complete "eventually", there is no mechanism to synchronize command execution with the completion (or even initiation) of these operations.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND
	28:23	**MI Command Opcode** Default Value: 21h MI_STORE_DATA_INDEX
	22	**Reserved**
	21	**Use Per-Process Hardware Status Page** Project: CHV, BSW If this bit is set, this command will index into the per-process hardware status page at offset 0K from the LRCA. If clear, the Global Hardware Status Page will be indexed. This bit must be '0' if the Execlist Enable bit is clear.
	20:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 1h Excludes DWord (0,1) = 1 for DWord, 2 for QWord Format: =n Total Length - 2
1	31:12	**Reserved** Project: All Format: MBZ

MI_STORE_DATA_INDEX

	11:2	**Offset** Project: All Format: U10 zero-based DWord offset into the HW status page. Format: HardwareStatusPageOffset[11:2]U32 This field specifies the offset (into the hardware status page) to which the data will be written. Note that the first few DWords of this status page are reserved for special-purpose data storage - targeting these reserved locations via this command is UNDEFINED.This address must be 8B aligned for a store "QW" command. Value: [16, 1023] / Name:
	1:0	**Reserved** Project: All Format: MBZ
2	31:0	**Data DWord 0** Project: All Format: U32 This field specifies the DWord value to be written to the targeted location.For a QWord write this DWord is the lower DWord of the QWord to be reported (DW 0).
3	31:0	**Data DWord 1** Project: All Format: U32 This field specifies the upper DWord value to be written to the targeted QWord location (DW 1).

MI_STORE_DATA_INDEX

MI_STORE_DATA_INDEX

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The MI_STORE_DATA_INDEX command requests a write of the data constant supplied in the packet to the specified offset from the System Address defined by the Hardware Status Page Address Register. As the write targets a System Address, the write operation is coherent with the CPU cache (i.e., the processor cache is snooped).

Programming Notes

- Use of this command with an invalid or uninitialized value in the Hardware Status Page Address Register is UNDEFINED.
- This command can be used for general software synchronization through variables in cacheable memory (i.e., where software does not need to poll uncached memory or device registers).
- This command simply initiates the write operation with command execution proceeding normally. Although the write operation is guaranteed to complete eventually, there is no mechanism to synchronize command execution with the completion (or even initiation) of these operations.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 21h MI_STORE_DATA_INDEX Format: OpCode
	22	**Reserved** Project: CHV, BSW
	21	**Use Per-Process Hardware Status Page** Project: CHV, BSW If this bit is set, this command will index into the per-process hardware status page at offset 0K from the LRCA. If clear, the Global Hardware Status Page will be indexed. This bit must be 0 if the Execlist Enable bit is clear.
	20:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 1h Format: =n Total Length - 2. Excludes DWord (0,1) = 1 for DWord, 2 for QWord.

MI_STORE_DATA_INDEX		
1	31:12	**Reserved** Format: MBZ
	11:2	**Offset** Format: U10 zero-based DWord offset into the HW status page. Format: HardwareStatusPageOffset[11:2]U32 This field specifies the offset (into the hardware status page) to which the data will be written. Note that the first few DWords of this status page are reserved for special-purpose data storage - targeting these reserved locations via this command is UNDEFINED. This address must be 8B aligned for a store QW command. Value: [16, 1023] — Name:
	1:0	**Reserved** Format: MBZ
2	31:0	**Data DWord 0** Format: U32 This field specifies the DWord value to be written to the targeted location.For a QWord write this DWord is the lower DWord of the QWord to be reported (DW 0).
3	31:0	**Data DWord 1** Format: U32 This field specifies the upper DWord value to be written to the targeted QWord location (DW 1).

MI_STORE_DATA_INDEX

MI_STORE_DATA_INDEX

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	2

The MI_STORE_DATA_INDEX command requests a write of the data constant supplied in the packet to the specified offset from the System Address defined by the Hardware Status Page Address Register. As the write targets a System Address, the write operation is coherent with the CPU cache (i.e., the processor cache is snooped).

Programming Notes

- Use of this command with an invalid or uninitialized value in the Hardware Status Page Address Register is UNDEFINED.
- This command can be used for general software synchronization through variables in cacheable memory (i.e., where software does not need to poll uncached memory or device registers).
- This command simply initiates the write operation with command execution proceeding normally. Although the write operation is guaranteed to complete "eventually", there is no mechanism to synchronize command execution with the completion (or even initiation) of these operations.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 21h MI_STORE_DATA_INDEX Format: OpCode
	22	**Reserved** Format: MBZ
	21	**Use Per-Process Hardware Status Page** Project: CHV, BSW If this bit is set, this command will index into the per-process hardware status page at offset 0K from the LRCA. If clear, the Global Hardware Status Page will be indexed. This bit must be '0' if the Execlist Enable bit is clear.
	20:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) = 2 for QWord Format: =n Total Length - 2

<table>
<tr><th colspan="3">MI_STORE_DATA_INDEX</th></tr>
<tr><td rowspan="3">1</td><td>31:12</td><td>Reserved
Format: MBZ</td></tr>
<tr><td>11:2</td><td>Offset
Format: U10 zero-based DWord offset into the HW status page
Format: GraphicsAddress[11:2]U32
This field specifies the offset (into the hardware status page) to which the data will be written. For a QWord write, the offset is valid down to bit 3 only.
<table><tr><th>Value</th><th>Name</th></tr><tr><td>[16, 1023]</td><td></td></tr></table>
<table><tr><th>Programming Notes</th></tr><tr><td>The first few DWords of this status page are reserved for special-purpose data storage - targeting these reserved locations via this command is UNDEFINED.</td></tr></table></td></tr>
<tr><td>1:0</td><td>Reserved
Format: MBZ</td></tr>
<tr><td>2</td><td>31:0</td><td>Data DWord 0
Format: U32 FormatDesc
This field specifies the upper DWord value to be written to the targeted QWord location (DW 1).</td></tr>
<tr><td>3</td><td>31:0</td><td>Data Word 1
Format: U32 FormatDesc
This field specifies the upper DWord value to be written to the targeted QWord location (DW 1).</td></tr>
</table>

MI_STORE_REGISTER_MEM

MI_STORE_REGISTER_MEM

Project:	CHV, BSW
Source:	CommandStreamer
Length Bias:	2

The MI_STORE_REGISTER_MEM command requests a register read from a specified memory mapped register location in the device and store of that DWord to memory. The register address is specified along with the command to perform the read.

Programming Notes	Source
• The command temporarily halts command execution. • The memory address for the write is snooped on the host bus. • This command should not be used from within a "non-privilege" batch buffer to access global virtual space. doing so will be treated as privilege access violation. Refer "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation. This command can be used within ring buffers and/or "privilege" batch buffers to access global virtual space. • This command will cause undefined data to be written to memory if given register addresses for the PGTBL_CTL_0 or FENCE registers.	
Source: BlitterCS, VideoCS, VideoEnhancementCS The source MMIO offset must be limited to any MMIO that is not replicated due to multiple slice configurations. If slice zero is disabled, then any MMIO read from this command streamer to a register replicated in the slice will cause a return value of zero.	BlitterCS, VideoCS, VideoEnhancementCS

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 24h MI_STORE_REGISTER_MEM Format: OpCode
	22	**Use Global GTT** Format: Boolean It is allowed for this bit to be set when executing this command from a privileged (secure) batch or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear. This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.

<table>
<tr><th colspan="3">MI_STORE_REGISTER_MEM</th></tr>
<tr><td></td><td>21</td><td>Reserved</td></tr>
<tr><td></td><td>21</td><td>Predicate Enable
Project: CHV, BSW
Source: RenderCS
Format: U1
If set, this command is executed (or not) depending on the current value of the MI Predicate internal state bit. This command is ignored only if PredicateEnable is set and the Predicate state bit is 0.</td></tr>
<tr><td></td><td>20:8</td><td>Reserved
Format: MBZ</td></tr>
<tr><td></td><td>7:0</td><td>DWord Length
Format: =n Total Length - 2

Value | Name | Project
2h | Excludes DWord (0,1) [Default] | CHV, BSW</td></tr>
<tr><td>1</td><td>31:23</td><td>Reserved
Format: MBZ</td></tr>
<tr><td></td><td>22:2</td><td>Register Address
Format: MMIOAddress[22:2]MMIO_Register
This field specifies Bits 22:2 of the Register offset the DWord will be read from. As the register address must be DWord-aligned, Bits 1:0 of that address MBZ.
Programming Notes
• Storing a VGA register is not permitted and will store an UNDEFINED value.
• The values of PGTBL_CTL0 or any of the FENCE registers cannot be stored to memory; UNDEFINED values will be written to memory if the addresses of these registers are specified.</td></tr>
<tr><td></td><td>1:0</td><td>Reserved</td></tr>
<tr><td>2..3
Project:
CHV, BSW</td><td>63:2</td><td>Memory Address
Project: CHV, BSW
Format: GraphicsAddress[63:2]MMIO
This field specifies the address of the memory location where the register value specified in the DWord above will be written. The address specifies the DWord location of the data.Range = GraphicsVirtualAddress[63:2] for a DWord register
GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].</td></tr>
<tr><td></td><td>1:0</td><td>Reserved</td></tr>
</table>

MI_STORE_URB_MEM

MI_STORE_URB_MEM	
Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The MI_STORE_URB_MEM command requests a URB read from a specified memory mapped URB location in the device and store of that DWord to memory. The URB address is specified along with the command to perform the read.

Programming Notes

- The command temporarily halts command execution.
- This command should not be used within a "non-secure" batch buffer to access global virtual space. Doing so will cause the command parser to perform the write with byte enables turned off. This command can be used within ring buffers and/or "secure" batch buffers.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 2Dh MI_STORE_URB_MEM Format: OpCode
	22:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2. Excludes DWord (0,1). Value: 2h / Name: **[Default]** / Project: CHV, BSW
1	31:15	**Reserved** Format: MBZ
	14:2	**URB Address** This field specifies Bits 14:2 of the URB offset the DWord will be read in the URB. This command only supports reading from the lower 32KB of the URB space.
	1:0	**Reserved** Format: MBZ

MI_STORE_URB_MEM		
2..3 **Project:** CHV, BSW	63:6	**Memory Address** <table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>GraphicsAddress[63:6]</td></tr></table>This field specifies the address of the location of where the value will be written to memory. The value must be in the first DW location of the cache line. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].
	5:0	**Reserved** <table><tr><td>Project:</td><td>CHV, BSW</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table>

MI_SUSPEND_FLUSH

MI_SUSPEND_FLUSH

Project:	CHV, BSW
Source:	VideoEnhancementCS
Length Bias:	1

Description	Project
Blocks PM Flush Requests.	CHV, BSW

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND
	28:23	**MI Command Opcode** Default Value: 0Bh MI_SUSPEND_FLUSH
	22:1	**Reserved** Project: All Format: MBZ
	0	**Suspend Flush** Project: All Format: Enable Description: This field suspends flush due to a PM flush request. (Project: CHV, BSW)

MI_SUSPEND_FLUSH

MI_SUSPEND_FLUSH

Project: CHV, BSW
Source: BlitterCS
Length Bias: 1

Description	Project
Blocks PM Flush Requests.	CHV, BSW

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND
	28:23	**MI Command Opcode** Default Value: 0Bh MI_SUSPEND_FLUSH
	22:1	**Reserved** Project: All Format: MBZ
	0	**Suspend Flush** Project: All Format: Enable Description: This field suspends flush due to a PM flush request. (Project: CHV, BSW)

MI_SUSPEND_FLUSH

MI_SUSPEND_FLUSH

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	1

Description	Project
Blocks PM Flush Requests.	CHV, BSW

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td>Command Type
Default Value: 0h MI_COMMAND
Format: OpCode</td></tr>
<tr><td></td><td>28:23</td><td>MI Command Opcode
Default Value: 0Bh MI_SUSPEND_FLUSH
Format: OpCode</td></tr>
<tr><td></td><td>22:1</td><td>Reserved
Format: MBZ</td></tr>
<tr><td></td><td>0</td><td>Suspend Flush
Format: Enable
<table><tr><th>Description</th><th>Project</th></tr><tr><td>This field suspends flush due to a PM flush request.</td><td>CHV, BSW</td></tr></table></td></tr>
</table>

MI_SUSPEND_FLUSH

MI_SUSPEND_FLUSH

Project:	CHV, BSW
Source:	VideoCS
Length Bias:	1

Description	Project
Blocks PM Flush Requests.	CHV, BSW

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND
	28:23	**MI Command Opcode** Default Value: 0Bh MI_SUSPEND_FLUSH
	22:1	**Reserved** Format: MBZ
	0	**Suspend Flush** Format: Enable Description: This field suspends flush due to a PM flush request. — Project: CHV, BSW

MI_TOPOLOGY_FILTER

MI_TOPOLOGY_FILTER

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	1

This command is used to specify a specific 3DPrimType value, where the CS will ignore all 3DPRIMITIVE commands that do no have a matching 3DPrimType. This primitive culling is optional (turned off by using this command with a Topology Filter Value of 0). **This command is specific to the Render command stream only.**

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 0Dh MI_TOPOLOGY_FILTER Format: OpCode
	22:6	**Reserved** Format: MBZ
	5:0	**Topology Filter Value** Format: 3D_Prim_Topo_Type [CHV, BSW] When non-zero, the CS will discard all 3DPRIMITIVE commands which do not match the specified 3DPrimTopologyType. When zero, no filtering is performed (normal operation).

MI_UPDATE_GTT

MI_UPDATE_GTT

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

The MI_UPDATE_GTT command is used to update GGTT page table entries in a coherent manner and at a predictable place in the command flow.

A MI_FLUSH_DWORD flush command with "CS Stall" bit set must be programmed prior to MI_UPDATE_GTT command, since work associated with preceding commands that are still in the pipeline may be referencing GTT entries that will be changed by its execution. The flush must also invalidate TLBs and read caches that may become invalid as a result of the changed GTT entries. A MI_FLUSH_DWORD flush command with "CS Stall" bit set must be programmed post MI_UPDATE_GTT command to ensure the GGTT is updated with modified page table entries before the following workload references the modified entries.

MI_FLUSH_DWORD flush is not required if it can be guaranteed that the pipeline is free of any work that relies on changing GTT entries (such as MI_UPDATE_GTT contained in a paging DMA buffer that is doing only update/mapping activities and no rendering).

MI_UPDATE_GTT command is privilege operation and will be converted to a no-op and an error flagged if it is executed from within a non-secure batch buffer.

PPGTT updates cannot be done via **MI_UPDATE_GTT**, gfx driver will have to use MI_STORE_DATA_IMM for PPGTT inline updates.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 23h MI_UPDATE_GTT Format: OpCode
	22:10	**Reserved**
	9:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:12	**Entry Address** Format: GraphicsAddress[31:12] This field holds the QW offset of the first table entry to be modified in GGTT.
	11:0	**Reserved**
2..n	31:0	**Entry Data** Format: PageTableEntry This Dword becomes the lower dword new page table entry. See PPGTT/Global GTT Table Entries (PTEs) in Memory Interface Registers.

MI_UPDATE_GTT

MI_UPDATE_GTT			
Project:	CHV, BSW		
Source:	RenderCS		
Length Bias:	2		

The MI_UPDATE_GTT command is used to update GTT page table entries in a coherent manner and at a predictable place in the command flow.
A PIPE_CONTROL flush command with "CS Stall" bit set must be programmed prior to MI_UPDATE_GTT command, since work associated with preceding commands that are still in the pipeline may be referencing GTT entries that will be changed by its execution. The flush must also invalidate TLBs and read caches that may become invalid as a result of the changed GTT entries. A PIPE_CONTROL flush command with "CS Stall" bit set must be programmed post MI_UPDATE_GTT command to ensure the GGTT is updated with modified page table entries before the following workload references the modified entries.
PIPE_CONTROL flush is not required if it can be guaranteed that the pipeline is free of any work that relies on changing GTT entries (such as MI_UPDATE_GTT contained in a paging DMA buffer that is doing only update/mapping activities and no rendering).
MI_UPDTE_GTT command is privilege operation and will be converted to a no-op and an error flagged if it is executed from within a non-secure batch buffer.
PPGTT updates cannot be done via **MI_UPDATE_GTT**, gfx driver will have to use MI_STORE_DATA_IMM for PPGTT inline updates.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 23h MI_UPDATE_GTT Format: OpCode
	22:10	**Reserved**
	9:0	**DWord Length** Default Value: 0h Format: =n Total Length - 2. Excludes DWord (0,1). **Programming Notes** The value of this field must not exceed a value 3Fh when programmed in a batch buffer with resource streamer enabled.
1	31:12	**Entry Address** Format: GraphicsAddress[31:12] This field holds the QW offset of the first table entry to be modified in GGTT.

MI_UPDATE_GTT		
	11:0	**Reserved**
2..n	63:0	**Entry Data** Format: PageTableEntry This Dword becomes the new page table entry. See PPGTT/Global GTT Table Entries (PTEs) in Memory Interface Registers.

MI_UPDATE_GTT

MI_UPDATE_GTT		
Project:	CHV, BSW	
Source:	VideoCS	
Length Bias:	2	

The MI_UPDATE_GTT command is used to update GGTT page table entries in a coherent manner and at a predictable place in the command flow.
A MI_FLUSH_DWORD flush command with "CS Stall" bit set must be programmed prior to MI_UPDATE_GTT command, since work associated with preceding commands that are still in the pipeline may be referencing GTT entries that will be changed by its execution. The flush must also invalidate TLBs and read caches that may become invalid as a result of the changed GTT entries. A MI_FLUSH_DWORD flush command with "CS Stall" bit set must be programmed post MI_UPDATE_GTT command to ensure the GGTT is updated with modified page table entries before the following workload references the modified entries.
MI_FLUSH_DWORD flush is not required if it can be guaranteed that the pipeline is free of any work that relies on changing GTT entries (such as MI_UPDATE_GTT contained in a paging DMA buffer that is doing only update/mapping activities and no rendering).
MI_UPDATE_GTT command is privilege operation and will be converted to a no-op and an error flagged if it is executed from within a non-secure batch buffer.
PPGTT updates cannot be done via **MI_UPDATE_GTT**, gfx driver will have to use MI_STORE_DATA_IMM for PPGTT inline updates.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 23h MI_UPDATE_GTT Format: OpCode
	22:10	**Reserved**
	9:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:12	**Entry Address** Format: GraphicsAddress[31:12] This field holds the QW offset of the first table entry to be modified in GGTT.
	11:0	**Reserved**
2..n	63:0	**Entry Data** Format: PageTableEntry This Dword becomes the lower dword new page table entry. See PPGTT/Global GTT Table Entries (PTEs) in Memory Interface Registers.

MI_UPDATE_GTT

MI_UPDATE_GTT

Project:	CHV, BSW
Source:	VideoEnhancementCS
Length Bias:	2

The MI_UPDATE_GTT command is used to update GGTT page table entries in a coherent manner and at a predictable place in the command flow.
A MI_FLUS_DWORD flush command with "CS Stall" bit set must be programmed prior to MI_UPDATE_GTT command, since work associated with preceding commands that are still in the pipeline may be referencing GTT entries that will be changed by its execution. The flush must also invalidate TLBs and read caches that may become invalid as a result of the changed GTT entries. A MI_FLUSH_DWORD flush command with "CS Stall" bit set must be programmed post MI_UPDATE_GTT command to ensure the GGTT is updated with modified page table entries before the following workload references the modified entries.
MI_FLUSH_DWORD flush is not required if it can be guaranteed that the pipeline is free of any work that relies on changing GTT entries (such as MI_UPDATE_GTT contained in a paging DMA buffer that is doing only update/mapping activities and no rendering).
MI_UPDTE_GTT command is privilege operation and will be converted to a no-op and an error flagged if it is executed from within a non-secure batch buffer.
PPGTT updates cannot be done via **MI_UPDATE_GTT**, gfx driver will have to use MI_STORE_DATA_IMM for PPGTT inline updates.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 23h MI_UPDATE_GTT Format: OpCode
	22:10	**Reserved**
	9:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:12	**Entry Address** Format: GraphicsAddress[31:12] This field holds the QW offset of the first table entry to be modified in GGTT.
	11:0	**Reserved**
2..n	63:0	**Entry Data** Format: PageTableEntry This Dword becomes the lower dword new page table entry. See PPGTT/Global GTT Table Entries (PTEs) in Memory Interface Registers.

MI_URB_ATOMIC_ALLOC

MI_URB_ATOMIC_ALLOC

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	1

This command is used to specify the region in URB allocated for URB atomic value storage.
This command is specific to the Render command stream only.

Programming Notes
This command can only be sent after a flush has occurred.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 09h MI_URB_ALLOC Format: OpCode
	22:20	**Reserved** Format: MBZ
	19:12	**URB Atomic Storage Offset** Format: U8 Number of 128B Entries This field specifies the offset of a 128B granular starting address in the URB. The value of **URB Atomic Storage Offset** plus the value of the **URB Atomic Storage Size** must not exceed 256. Value: [0,255] — Name: — Description: 0-(32KB-128B)
	11:9	**Reserved** Format: MBZ
	8:0	**URB Atomic Storage Size** Format: U9 Number of 128B Entries This field specifies the size of the buffer in the URB in number of 128B entries. If this field has a value of zero then the URB Atomic allocation is disabled and will not be context save/restored. Value: [0,256] — Name: — Description: 0-32KB

MI_URB_CLEAR

MI_URB_CLEAR

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	2

The MI_URB_CLEAR command allows SW to clear (write zero) to a section in the URB.

Programming Notes

- The command temporarily halts command execution.
- This command is part of context save/restore. Only the last instance will be part of context.
- This command requires the 3D pipeline to be flushed before execution.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 19h MI_URB_CLEAR Format: OpCode
	22:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Format: =n Total Length - 2. Excludes DWord (0,1).
1	31:30	**Reserved**
	29:16	**URB Clear Length** Project: CHV, BSW This field specifies the number of 256b entries in the URB to be cleared to zero. Value: [0,16383] — Name:
	15	**Reserved**
	14:0	**URB Address** Project: CHV, BSW Format: URBAddress[19:5] 256b aligned This field specifies Bits 19:5 of the URB Address

MI_USER_INTERRUPT

MI_USER_INTERRUPT		
Project:	CHV, BSW	
Source:	VideoEnhancementCS	
Length Bias:	1	

The MI_USER_INTERRUPT command is used to generate a User Interrupt condition. The parser will continue parsing after processing this command. See User Interrupt.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 02h MI_USER_INTERRUPT Format: OpCode
	22:0	**Reserved** Project: All Format: MBZ

MI_USER_INTERRUPT

MI_USER_INTERRUPT		
Project:	CHV, BSW	
Source:	BlitterCS	
Length Bias:	1	
The MI_USER_INTERRUPT command is used to generate a User Interrupt condition. The parser will continue parsing after processing this command. See User Interrupt.		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 0h MI_COMMAND
	28:23	**MI Command Opcode** Default Value: 02h MI_USER_INTERRUPT
	22:0	**Reserved** Project: All Format: MBZ

MI_USER_INTERRUPT

MI_USER_INTERRUPT		
Project:	CHV, BSW	
Source:	RenderCS	
Length Bias:	1	
The MI_USER_INTERRUPT command is used to generate a User Interrupt condition. The parser will continue parsing after processing this command. See User Interrupt.		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 02h MI_USER_INTERRUPT Format: OpCode
	22:0	**Reserved** Format: MBZ

MI_USER_INTERRUPT

MI_USER_INTERRUPT		
Project:	CHV, BSW	
Source:	VideoCS	
Length Bias:	1	
The MI_USER_INTERRUPT command is used to generate a User Interrupt condition. The parser will continue parsing after processing this command. See User Interrupt.		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 02h MI_USER_INTERRUPT Format: OpCode
	22:0	**Reserved** Format: MBZ

MI_WAIT_FOR_EVENT

MI_WAIT_FOR_EVENT

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	1

The MI_WAIT_FOR_EVENT command is used to pause command stream processing until a specific event occurs or while a specific condition exists. Only one event/condition can be specified -- specifying multiple events is UNDEFINED.The effect of the wait operation depends on the source of the command. If executed from a batch buffer, the parser will halt (and suspend command arbitration) until the event/condition occurs. If executed from a ring buffer, further processing of that ring will be suspended, although command arbitration (from other rings) will continue. Note that if a specified condition does not exist (the condition code is inactive) at the time the parser executes this command, the parser proceeds, treating this command as a no-operation.If execution of this command from a primary ring buffer causes a wait to occur, the active ring buffer will effectively give up the remainder of its time slice (required in order to enable arbitration from other primary ring buffers).

Programming Notes
[Ring Buffer Mode Of scheduling Only][Blitter CS]: HW loses Page Directory (PPGTT) information on becoming IDLE. SW must always program the PD information following MI_WAIT_FOR_EVENT command. This will ensure Page Directory information gets reprogrammed on exiting IDLE flow triggered on MI_WAIT_FOR_EVENT. Alternatively SW can disable IDLE flows on MI_WAIT_FOR_EVENT by setting below bits in “BCS_ECOSKPD” register. Disable GT C6 Enter Due to Blitter Waiting on Vblank Disable GT C6 Enter Due to Blitter Waiting on Scanline Disable GT C6 Enter Due to Blitter Waiting on Flip Done

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND
	28:23	**MI Command Opcode** Default Value: 03h MI_WAIT_FOR_EVENT
	22	**Reserved**
	21	**Display Pipe C Vertical Blank Wait Enable** Project: CHV, BSW Format: Enable This field enables a wait until the next Display Pipe C "Vertical Blank" event occurs. This event is described as the start of the next Display C vertical blank period. Note that this can cause a wait for up to an entire refresh period. See Vertical Blank Event in the Device Programming Interface chapter of MI Functions. **Programming Notes** If this bit is set, then RC6 cannot occur while waiting for this event to complete. SW must disable IDLE sequence or RC6 prior to the command. IDLE sequence and RC6 must be reenabled after the command to allow the hardware to enter RC6.

MI_WAIT_FOR_EVENT

	20	**Display Sprite C Flip Pending Wait Enable** Project: All Format: Enable This field enables a wait for the duration of a Display Sprite C "Flip Pending" condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). **Programming Notes** If this bit is set, then RC6 cannot occur while waiting for this event to complete. SW must disable IDLE sequence or RC6 prior to the command. IDLE sequence and RC6 must be reenabled after the command to allow the hardware to enter RC6.
	19	**Display Sprite C3 Flip Pending Wait Enable** Project: CHV, BSW Format: Enable This field enables a wait for the duration of a Display Sprite C3 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).
	18	**Display Sprite B3 Flip Pending Wait Enable** Project: CHV, BSW Format: Enable This field enables a wait for the duration of a Display Sprite B3 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).
	17	**Display Sprite A3 Flip Pending Wait Enable** Project: CHV, BSW Format: Enable This field enables a wait for the duration of a Display Sprite A3 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).
	16	**Display Sprite C2 Flip Pending Wait Enable** Project: CHV, BSW Format: Enable This field enables a wait for the duration of a Display Sprite C2 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).
	15	**Display Plane C Flip Pending Wait Enable** Project: All Format: Enable

MI_WAIT_FOR_EVENT

This field enables a wait for the duration of a Display Plane C "Flip Pending" condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).

Programming Notes
If this bit is set, then RC6 cannot occur while waiting for this event to complete. SW must disable IDLE sequence or RC6 prior to the command. IDLE sequence and RC6 must be reenabled after the command to allow the hardware to enter RC6.

14 — Display Pipe C Scan Line Wait Enable

Project:	CHV, BSW
Format:	Enable

This field enables a wait while a Display Pipe C "Scan Line" condition exists. This condition is defined as the the start of the scan line specified in the Pipe C Display Scan Line Count Range Compare Register.

Programming Notes
If this bit is set, then RC6 cannot occur while waiting for this event to complete. SW must disable IDLE sequence or RC6 prior to the command. IDLE sequence and RC6 must be reenabled after the command to allow the hardware to enter RC6.

13:12 — Reserved

Project:	All
Format:	MBZ

11 — Display Pipe B Vertical Blank Wait Enable

Project:	CHV, BSW
Format:	Enable

This field enables a wait until the next Display Pipe B "Vertical Blank" event occurs. This event is described as the start of the next Display Pipe B vertical blank period. Note that this can cause a wait for up to an entire refresh period.

10 — Display Sprite B Flip Pending Wait Enable

Project:	All
Format:	Enable

This field enables a wait for the duration of a Display Sprite B "Flip Pending" condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).

9 — Display Plane B Flip Pending Wait Enable

Project:	All
Format:	Enable

This field enables a wait for the duration of a Display Plane B "Flip Pending" condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).

MI_WAIT_FOR_EVENT

	8	**Display Pipe B Scan Line Wait Enable** Project: CHV, BSW Format: Enable This field enables a wait while a Display Pipe B "Scan Line" condition exists. This condition is defined as the the start of the scan line specified in the Pipe B Display Scan Line Count Range Compare Register. **Programming Notes** If this bit is set, then RC6 cannot occur while waiting for this event to complete. SW must disable IDLE sequence or RC6 prior to the command. IDLE sequence and RC6 must be reenabled after the command to allow the hardware to enter RC6.
	7	**Display Sprite B2 Flip Pending Wait Enable** Project: CHV, BSW Format: Enable This field enables a wait for the duration of a Display Sprite B2 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).
	6	**Display Sprite A2 Flip Pending Wait Enable** Project: CHV, BSW Format: Enable This field enables a wait for the duration of a Display Sprite A2 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).
	5:4	**Reserved** Project: All Format: MBZ
	3	**Display Pipe A Vertical Blank Wait Enable** Project: CHV, BSW Format: Enable This field enables a wait until the next Display Pipe A "Vertical Blank" event occurs. This event is described as the start of the next Display Pipe A vertical blank period. Note that this can cause a wait for up to an entire refresh period.
	2	**Display Sprite A Flip Pending Wait Enable** Project: All Format: Enable This field enables a wait for the duration of a Display Sprite A "Flip Pending" condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).

MI_WAIT_FOR_EVENT

	1	**Display Plane A Flip Pending Wait Enable** Project: All Format: Enable This field enables a wait for the duration of a Display Plane A "Flip Pending" condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).
	0	**Display Pipe A Scan Line Wait Enable** Project: CHV, BSW Format: Enable This field enables a wait while a Display Pipe A "Scan Line" condition exists. This condition is defined as the the start of the scan line specified in the Pipe A Display Scan Line Count Range Compare Register. **Programming Notes** If this bit is set, then RC6 cannot occur while waiting for this event to complete. SW must disable IDLE sequence or RC6 prior to the command. IDLE sequence and RC6 must be reenabled after the command to allow the hardware to enter RC6.

MI_WAIT_FOR_EVENT

MI_WAIT_FOR_EVENT

Project:	CHV, BSW
Source:	RenderCS
Length Bias:	1

Description
The MI_WAIT_FOR_EVENT command is used to pause command stream processing **of this pipe only** until a specific event occurs or while a specific condition exists. See Wait Events/Conditions, Device Programming Interface in *MI Functions*. Only one event/condition can be specified. Specifying multiple events is UNDEFINED. Once parsed, the parser will halt (and suspend command arbitration) until the event/condition occurs. Note that if a specified condition does not exist (the condition code is inactive) at the time the parser executes this command, the parser proceeds, treating this command as a no-operation. If CSunit is waiting for V-blank or flip done, HW can go into RC1/RC6 state. MI_NOOP setting NOP register (or any other benign command) must be set after MI_WAIT_FOR_EVENT under the following conditions: • Back-to-back MI_WAIT_FOR_EVENT commands • MI_WAIT_FOR_EVENT is the last command before head = tail
Events must be unmasked in the Display Engine Render Response Mask Register (DE RRMR 0x44050) prior to waiting for them with a MI_WAIT_FOR_EVENT command, or in the case of flips or scanlines, prior to starting the flip or loading the scanline. Unmasked events will wake command streamer as they occur, so for improved power savings it is recommended to only unmask events that are required. Programming the DE RRMR register can be done through MMIO or a LOAD_REGISTER_IMMEDIATE command.
Execution List Mode of Scheduling: CS on evaluating MI_WAIT_FOR_EVENT to be unsuccessful (has to wait for event to happen) triggers synchronous context switch stating the switch reason in Context Status Buffer. Note that synchronous context switch can be inhibited through programming "**Inhibit Synchronous Context Switch**" bit in CTXT_SR_CTL register or by disabling arbitration through MI_ARB_ON_OFF command.

Programming Notes
Ring Buffer Mode of Scheduling Only: SW must always program a dummy **MI_SEMAPHORE_WAIT** command in **Signal Mode** which is always successful prior to programming MI_WAIT_FOR_EVENT. If the above programming restriction is not followed, in certain order of programming sequences HW would enter IDLE_DOP instead of IDLE_C6 on encountering MI_WAIT_FOR_EVENT unsuccessful.
Render CS Only: SW must always program PIPE_CONTROL with "CS Stall" and "Render Target Cache Flush Enable" set prior to programming MI_WAIT_FOR_EVENT command for GPGPU workloads i.e when pipeline select is GPGPU via PIPELINE_SELECT command. This is required to achieve better GPGPU preemption latencies for certain programming sequences. If programming PIPE_CONTROL has performance implications then preemption latencies can be trade off against performance by not implementing this programming note.
Execlist Mode of Scheduling Only: When MI_WAIT_FOR_EVENT command results in context switch on "Wait On V-blank", "Display Plane" field indicates the Display pipe for which the wait for event was un-successful.

MI_WAIT_FOR_EVENT

When a context switched out due to "Wait on V-blank" is resubmitted by scheduler without waiting for the corresponding V-blank event to be satisfied, it might happen that the context can get switched out again by HW due to V-blank not satisfied, when this condition occurs due to known HW issue "Display Plane" field is not indicated correctly. To work around this issue SW must do one of the below

1. Always program MI_WAIT_FOR_EVENT for V-blank with Inhibit Synchronous Context Switch Set so that a context never gets switched out with context switch reason as only due to "Wait On V-blank". However a context waiting for V-blank event to be satisfied can get switched out due to preemption on submission of a pending execlist. In this case scheduler can resubmit the switched out context at appropriate time similar to any other preempted context. **OR**
2. When MI_WAIT_FOR_EVENT for V-blank is programmed with Inhibit Synchronous Context Switch Reset. Scheduler on detecting a context switched out due to Wait On V-blank must not resubmit the context unless the corresponding V-blank event is satisfied. **OR**
3. Scheduler must maintain its own record of the Pipe assigned to a given context for Wait on V-blank and must not look at the "Display Plane" field reported in the context swtich status.

Execlist Mode of Scheduling: CS takes couple of clocks to parse MI_WAIT_FOR_EVENT command and in a specific clock while parsing it makes the decision to do Synchronous Context Switch if the event is unsuccessful. When CS receives Display message satisfying the MI_WAIT_FOR_EVENT on the same clock when it makes decision to do synchronous context switch. Only way this issue can be avoided is by inhibiting synchronous context switch on MI_WAIT_FOR_EVENT command. Synchronous context switch on MI_WAIT_FOR_EVENT can be inhibited by setting "**Inhibit Synchronous Context Switch**" bit in CTXT_SR_CTL register through MI_LOAD_REGISTER_IMM command prior to MI_WAIT_FOR_EVENT command and resetting the same after MI_WAIT_FOR_EVENT command.

Ex:

MI_LOAD_REGISTER_IMM (CTXT_SR_CTL, Inhibit Synchronous Context Switch set)
MI_WAIT_FOR_EVENT
MI_LOAD_REGISTER_IMM (CTXT_SR_CTL, Inhibit Synchronous Context Switch reset)

When the above sequence has to be programmed in a non-privileged batch buffer, SW has to ensure CTXT_SR_CTL register is forced to non-privileged register through one of the FORCE_TO_NONPRIV registers.

DWord	Bit	Description	
0	31:29	**Command Type**	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	**MI Command Opcode**	
		Default Value:	03h MI_WAIT_FOR_EVENT
		Format:	OpCode
	22	**Reserved**	

MI_WAIT_FOR_EVENT

	21	**Display Pipe C Vertical Blank Wait Enable** Project: CHV, BSW Format: Enable **Description** This field enables a wait until the next Display Pipe C Vertical Blank event occurs. This event is described as the start of the next Display C vertical blank period. Note that this can cause a wait for up to an entire refresh period. **Render and Blitter Engines**
	20	**Display Sprite C Flip Pending Wait Enable** Project: CHV, BSW Format: Enable **Description** This field enables a wait for the duration of a Display Sprite C Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). **Render and Blitter Engines**
	19	**Display Sprite C3 Flip Pending Wait Enable** Project: CHV, BSW Format: Enable This field enables a wait for the duration of a Display Sprite C3 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).
	18	**Display Sprite B3 Flip Pending Wait Enable** Project: CHV, BSW Format: Enable This field enables a wait for the duration of a Display Sprite B3 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).
	17	**Display Sprite A3 Flip Pending Wait Enable** Project: CHV, BSW Format: Enable This field enables a wait for the duration of a Display Sprite A3 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).

MI_WAIT_FOR_EVENT

	16	**Display Sprite C2 Flip Pending Wait Enable** Project: CHV, BSW Format: Enable This field enables a wait for the duration of a Display Sprite C2 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).
	15	**Display Plane C Flip Pending Wait Enable** Project: CHV, BSW Format: Enable **Description** This field enables a wait for the duration of a Display Plane C "Flip Pending" condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). **Render and Blitter Engines**
	14	**Display Pipe C Scan Line Wait Enable** Project: CHV, BSW Format: Enable **Description** This field enables a wait while a Display Pipe C Scan Line condition exists. This condition is defined as the start of the scan line specified in the Pipe C Display Scan Line Count Range Compare Register. **Render and Blitter Engines**
	13	**Reserved** Project: CHV, BSW Format: MBZ
	12	**Reserved**
	11	**Display Pipe B Vertical Blank Wait Enable** Format: Enable **Description** This field enables a wait until the next Display Pipe B "Vertical Blank" event occurs. This event is described as the start of the next Display Pipe B vertical blank period. Note that this can cause a wait for up to an entire refresh period. **Render and Blitter Engines**

MI_WAIT_FOR_EVENT		
	10	**Display Sprite B Flip Pending Wait Enable** Format: Enable **Description** This field enables a wait for the duration of a Display Sprite B "Flip Pending" condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). **Render and Blitter Engines**
	9	**Display Plane B Flip Pending Wait Enable** Format: Enable **Description** This field enables a wait for the duration of a Display Plane B Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers. **Render and Blitter Engines**
	8	**Display Pipe B Scan Line Wait Enable** Format: Enable **Description** This field enables a wait while a Display Pipe B Scan Line condition exists. This condition is defined as the start of the scan line specified in the Pipe B Display Scan Line Count Range Compare Register. **Render and Blitter Engines**
	7	**Display Sprite B2 Flip Pending Wait Enable** Project: CHV, BSW Format: Enable This field enables a wait for the duration of a Display Sprite B2 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).
	6	**Display Sprite A2 Flip Pending Wait Enable** Project: CHV, BSW Format: Enable This field enables a wait for the duration of a Display Sprite A2 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).
	5	**Reserved**
	4	**Reserved**

MI_WAIT_FOR_EVENT		
	3	**Display Pipe A Vertical Blank Wait Enable** Format: Enable **Description** This field enables a wait until the next Display Pipe A "Vertical Blank" event occurs. This event is described as the start of the next Display Pipe A vertical blank period. Note that this can cause a wait for up to an entire refresh period. **Render and Blitter Engines**
	2	**Display Sprite A Flip Pending Wait Enable** Format: Enable **Description** This field enables a wait for the duration of a Display Sprite A "Flip Pending" condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). **Render and Blitter Engines**
	1	**Display Plane A Flip Pending Wait Enable** Format: Enable **Description** This field enables a wait for the duration of a Display Plane A "Flip Pending" condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). **Render and Blitter Engines**
	0	**Display Pipe A Scan Line Wait Enable** Format: Enable **Description** This field enables a wait while a Display Pipe A "Scan Line" condition exists. This condition is defined as the start of the scan line specified in the Pipe A Display Scan Line Count Range Compare Register. **Render and Blitter Engines**

Move

mov - Move

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The mov instruction moves the components in src0 into the channels of dst. If src0 and dst are of different types, format conversion is performed. If src0 is a scalar immediate, the immediate value is loaded into enabled channels of dst. A mov with the same source and destination type, no source modifier, and no saturation is a raw move. A packed byte destination region (B or UB type with HorzStride == 1 and ExecSize > 1) can only be written using raw move.

When denorm mode is flush to zero, a raw mov instruction with saturation modifier will not flush the denorm input or output to zero (Denorm is preserved).

Format: [(pred)] mov[.cmod] (exec_size) dst src0

Programming Notes
A *mov* instruction with a source modifier always copies a denorm source value to a denorm destination value (in the manner of a raw move).
There is no direct conversion from B/UB to DF or DF to B/UB. Use two instructions and a word or DWord intermediate type.
There is no direct conversion from B/UB to Q/UQ or Q/UQ to B/UB. Use two instructions and a word or DWord intermediate integer type.
There is no direct conversion from HF to DF or DF to HF. Use two instructions and F (Float) as an intermediate type.
There is no direct conversion from HF to Q/UQ or Q/UQ to HF. Use two instructions and F (Float) or a word integer type or a DWord integer type as an intermediate type.

Restriction
Raw move is not supported for Float values in ALT mode if any values are infinities or NaNs.
An accumulator can be a source or destination operand but not both.

Syntax
[(pred)] mov[.cmod] (exec_size) reg reg [(pred)] mov[.cmod] (exec_size) reg imm32 [(pred)] mov[.cmod] (exec_size) reg imm64

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = src0.chan[n]; } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

mov - Move

Src Types	Dst Types	Project
*B,*W,*D	*B,*W,*D	
*B,*W,*D	F	
F	*B,*W,*D	
F	F	
*W,*D	DF	CHV, BSW
F	DF	CHV, BSW
DF	*W,*D	CHV, BSW
DF	F	CHV, BSW
DF	DF	CHV, BSW
*W,*D,*Q	*W,*D,*Q	CHV, BSW
F	*Q	CHV, BSW
DF	*Q	CHV, BSW
*Q	F	CHV, BSW
*Q	DF	CHV, BSW
*B,*W,*D	HF	CHV, BSW
F	HF	CHV, BSW
HF	*B,*W,*D	CHV, BSW
HF	F	CHV, BSW
HF	HF	CHV, BSW

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32 [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Move Indexed

movi - Move Indexed

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The movi instruction performs a fast component-wise indexed move for subfields from src0 to dst. The source operand must be an indirectly-addressed register. All channels of the source operand share the same register number, which is provided by the register field of the first address subregister, with a possible immediate register offset. The register fields of the subsequent address subregisters are ignored by hardware. The subregister number of a source channel is provided by the subregister field of the corresponding address subregister, with a possible immediate subregister offset.
The destination register may be either a directly-addressed or an indirectly-addressed register.
This instruction effectively performs a subfield shuffling from one register to another. Up to eight subfields can be selected by an instruction.

Format: [(pred)] movi (exec_size) dst src0 src1

Programming Notes
HW Implementation Details: The source register is calculated by adding the register portion of the first index register with the register portion of the address immediate, a0.0[11:5] + addr_imm[9:5] For byte movi, byte0 of the destination is selected by (a0.0[4:0]), byte1 is selected by (a0.1[4:0]), ..., and byte7 is selected by (a0.7[4:0]). The rest of the bytes are undefined. For word movi, byte0 of the destination is selected by (a0.0[4:1] & 0), byte1 is selected by (a0.0[4:1] & 1), byte2 is selected by (a0.1[4:1] & 0), byte3 is selected by (a0.1[4:1] & 1), ..., and byte15 is selected by (a0.7[4:1] & 1). The rest of the bytes are undefined. For DWord or float movi, byte0 of the destination is selected by (a0.0[4:2] & 00b), byte1 is selected by (a0.0[4:2] & 01b), byte2 is selected by (a0.0[4:2] & 10b), byte3 is selected by (a0.0[4:2] & 11b), byte4 is selected by (a0.1[4:2] & 00b), byte5 is selected by (a0.1[4:2] & 01b), ..., byte31 is selected by (a0.7[4:2] & 11b). For all 3 conditions above, a0.n[4:0] = a0.n[4:0] + addr_imm[4:0].

Restriction
Source operand cannot be accumulators. The source operand must be a general register.
The source and destination must have the same type.
The execution size must be 8.
The address register for the source must be aligned to the base (a0.0).
The destination register (directly or indirectly addressed) must be 16-byte aligned.
The destination region (directly or indirectly addressed) must point to the same GRF register.
The destination stride in bytes must equal the source element size in bytes.
The Align16 access mode is not allowed.
All the index registers (address subregisters) used must point to the same GRF register.
The instruction must use 1x1 indirect regioning.

movi - Move Indexed

The destination offset is only used to create channel enables. Each element of the destination is directly mapped to the index registers for the movi instruction. i.e. a0.0 -> dst.0, a0.1 -> dst.1, a0.2 -> dst.2, etc.
Only 8 address subregisters are used (a0.0-a0.7). Destination element 8 will be sourced from address register zero (a0.0), dst.9 <-a0.1, etc. This is an exception to the above restriction, for example: movi (8) r31.8:uw r[a0.0,0]<8;8,1>:uw // r31.8:uw<-a0.0:uw, r31.9:uw<-a0.1:uw, etc.
Conditional Modifier is not allowed for this instruction.

Syntax
[(pred)] movi (exec_size) reg reg imm

Pseudocode
Evaluate(WrEn); srcregfile = regfile(src0); srcregbase = reg(address[0]) + reg(addr_imm); for (n = 0; n < RegWidth; n++) { if (WrEn.chan[n]) { srcsubreg = subreg(address[n] + addr_imm); dst.chan[n] = srcregfile.srcreg.srcsubreg; } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	Y	Y

Src Types	Dst Types
B	B
UB	UB
W	W
UW	UW
D	D
UD	UD
F	F

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32 [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Multiply

mul - Multiply

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The mul instruction performs component-wise multiplication of src0 and src1 and stores the results in dst. When multiplying integer datatypes, if src0 is DW and src1 is W, irrespective of the destination datatype, the accumulator maintains full 48-bit precision. This is required to handle the macro for 32x32 multiplication. The macro described in the mach instruction should be used to obtain the full precision 64-bit multiplication results. Note: A 32x32 multiply operation is handled natively, without a macro. When operating in this mode, the resulting 64-bit data is packed, unlike the macro, where the lower and upper 32 bits of the result are written to different general registers by two separate instructions. Refer to the macro description for details. When multiplying integer data types, if one of the sources is a DW, the resulting full precision data is stored in the accumulator. However, if the destination data type is either W or DW, the low bits of the result are written to the destination register and the remaining high bits are discarded. This results in undefined Overflow and Sign flags. Therefore, conditional modifiers and saturation (.sat) cannot be used in this case.

Format: [(pred)] mul[.cmod] (exec_size) dst src0 src1

Restriction
Integer source operands cannot be accumulators.
When multiplying a DW and any lower precision integer, the DW operand must on src0.
[CHV, BSW-A]: DW * W is not supported outside MACH macro.
[CHV, BSW]: When multiplying DW x DW, the dst cannot be accumulator.

Syntax
[(pred)] mul[.cmod] (exec_size) reg reg reg [(pred)] mul[.cmod] (exec_size) reg reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = src0.chan[n] * src1.chan[n]; } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types	Project
*B	*B	
*B	*W	
*B	*D	
*W	*W	
*W	*D	
*W,*D	*D	

mul - Multiply

*D	*Q	
F	F	
DF	DF	
HF	HF	
HF, F	HF, F	CHV, BSW

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Multiply Accumulate

mac - Multiply Accumulate

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The mac instruction takes component-wise multiplication of src0 and src1, adds the results with the corresponding accumulator values, and then stores the final results in dst.

Format: [(pred)] mac[.cmod] (exec_size) dst src0 src1

Programming Notes
When source and destination datatypes are different, the implied datatype for the accumulator operand is always the destination datatype.

Restriction
Accumulator is an implicit source and thus cannot be an explicit source operand.

Syntax
[(pred)] mac[.cmod] (exec_size) reg reg reg [(pred)] mac[.cmod] (exec_size) reg reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = src0.chan[n] * src1.chan[n] + acc0.chan[n]; } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types	Project
*B,*W	*B,*W,*D	
F	F	
DF	DF	CHV, BSW
HF	HF	CHV, BSW
HF, F	HF, F	CHV, BSW

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM [CHV, BSW]

mac - Multiply Accumulate		
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Multiply Accumulate High

mach - Multiply Accumulate High

Project:	CHV, BSW
Source:	Eulsa
Length Bias:	4

The mach instruction performs DWord integer multiply-accumulate operation and outputs the high DWord (bits 63:32). For each enabled channel, this instruction multiplies the DWord in src0 with the high word of the DWord in src1, left shifts the result by 16 bits, adds it with the corresponding accumulator values, and keeps the whole 64-bit result in the accumulator. It then stores the high DWord (bits 63:32) of the results in dst. This instruction is intended to be used to emulate 32-bit DWord integer multiplication by using the large number of bits available in the accumulator. For example, the following instructions perform vector multiplication of two 32-bit signed integer sources from r2 and r3 and store the resulting vectors with the high 32 bits in r5 and the low 32 bits in r6. mul (8) acc0:d r2.0<8;8,1>:d r3.0<16;8,2>:uw mach (8) r5.0<1>:d r2.0<8;8,1>:d r3.0<8;8,1>:d mov (8) r6.0<1>:d acc0:d // Low 32 bits. Here is a different example including negation. An added preliminary mov is required for source modification on src1. mov (8) r3.0<1>:d -r3<8;8,1>:d mul (8) acc0:d r2.0<8;8,1>:d r3.0<16;8,2>:uw mach (8) r5.0<1>:d r2.0<8;8,1>:d r3.0<8;8,1>:d // High 32 bits mov (8) r6.0<1>:d acc0:d // Low 32 bits. The mach should have channel enable from the destHI of IMUL, the mov should have the channel enable from the destLO of IMUL. As mach is used to generate part of the 64-bit DWord integer results, saturation modifier should not be used. In fact, saturation modifier should not be used for any of these four instructions. Source and destination operands must be DWord integers. Source and destination must be of the same type, signed integer or unsigned integer. If dst is UD, src0 and src1 may be UD and/or D. However, if any of src0 and src1 is D, source modifier (abs) must be present to convert it to match with dst. If dst is D, src0 and src1 must also be D. They cannot be UD as it may cause unexpected overflow because the computed results are limited to 64 bits.

Format: [(pred)] mach[.cmod] (exec_size) dst src0 src1

Restriction
Accumulator is an implicit source and thus cannot be an explicit source operand.
AccWrEn is required.

Syntax
[(pred)] mach[.cmod] (exec_size) reg reg reg [(pred)] mach[.cmod] (exec_size) reg reg imm32

Pseudocode

Evaluate(WrEn);

```
 for ( n = 0; n < exec_size; n++ ) { if ( WrEn.chan[n] ) { acc.chan[n][63:0] =
(src1.chan[n][31:16] * src0.chan[n][31:0]) « 16 + acc.chan[n][63:0];
dst.chan[n][31:0] = acc.chan[n][63:32]; } }
```

Errata	Description
	A source modifier must not be used on src1 for the macro operation. This applies to both mul and mach of the macro. If source modifier is required, an additional mov instruction may be used before the macro.

mach - Multiply Accumulate High

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	Y	Y

Src Types	Dst Types
D	D
UD	UD

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Multiply Add

mad - Multiply Add

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The mad instruction takes component-wise multiplication of src1 and src2, adds the results with the corresponding src0 values, and then stores the final results in dst.

Format: [(pred)] mad[.cmod] (exec_size) dst src0 src1 src2

Restriction	Project
[CHV, BSW]: No explicit accumulator access because this is a three-source instruction. AccWrEn is allowed for implicitly updating the accumulator.	CHV, BSW
[CHV, BSW]: All three-source instructions have certain restrictions, described in Instruction Formats [CHV, BSW].	CHV, BSW

Syntax
[(pred)] mad[.cmod] (exec_size) reg reg reg reg

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = src1.chan[n] * src2.chan[n] + src0.chan[n]; } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types	Project
F	F	
DF	DF	CHV, BSW
HF	HF	CHV, BSW
HF, F	HF, F	CHV, BSW

DWord	Bit	Description
0..3	127:126	**Reserved** Format: MBZ
	125:106	**Source 2** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC [CHV, BSW]
	105	**Reserved** Format: MBZ
	104:85	**Source 1** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC [CHV, BSW]

mad - Multiply Add

	84	**Reserved** Format: MBZ
	83:64	**Source 0** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC [CHV, BSW]
	63:56	**Destination Register Number** Format: DstRegNum [CHV, BSW]
	55:53	**Destination Subregister Number** Format: DstSubRegNum[2:0]
	52:49	**Destination Channel Enable** Format: ChanEn[4] Four channel enables are defined for controlling which channels are written into the destination region. These channel mask bits are applied in a modulo-four manner to all ExecSize channels. There is 1-bit Channel Enable for each channel within the group of 4. If the bit is cleared, the write for the corresponding channel is disabled. If the bit is set, the write is enabled. Mnemonics for the bit being set for the group of 4 are x, y, z, and w, respectively, where x corresponds to Channel 0 in the group and w corresponds to channel 3 in the group
	48:42	**Reserved** Format: MBZ
	41:40	**Source 2 Modifier** Exists If: //([Property[Source Modifier]=='true') Format: SrcMod [CHV, BSW]
	39:38	**Source 1 Modifier** Exists If: //([Property[Source Modifier]=='true') Format: SrcMod [CHV, BSW]
	41:36	**Reserved** Exists If: //([Property[Source Modifier]=='false') Format: MBZ
	37:36	**Source 0 Modifier** Exists If: //([Property[Source Modifier]=='true') Format: SrcMod [CHV, BSW]
	35	**Reserved** Format: MBZ
	34	**Reserved** Format: MBZ
	33	**Flag Subregister Number** This field contains the flag subregister number for instructions with a non-zero Conditional Modifier.

mad - Multiply Add		
	32	**Reserved** Format: MBZ
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Multiply Add for Macro

madm - Multiply Add for Macro

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The madm instruction takes component-wise multiplication of src1 and src2, adds the results with the corresponding src0 values, and then stores the final results in dst. The source and destination operands have a higher precision carried in the exponent for this operation. The madm instruction is used for macro operations, where precision is accumulated over several instructions. This accumulation requires the exponent to increase by 2 extra bits across multiple madm operations. Refer to Macros Defined in 'Math' Section for usage and restrictions of this operation.

Format: [(pred)] madm[.cmod] (exec_size) dst src0 src1 src2

Restriction
Accumulator access is restricted to the special accumulators (acc2-acc9). Refer to the Accumulator Section for details on the special accumulators.

Syntax
[(pred)] madm[.cmod] (exec_size) reg reg reg reg

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = src1.chan[n] * src2.chan[n] + src0.chan[n]; } }

Predication	Conditional Modifier	Saturation	Source Modifier
N	N	N	N

Src Types	Dst Types
F	F
DF	DF

DWord	Bit	Description
0..3	127:126	**Reserved** Format: MBZ
	125:106	**Source 2** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC [CHV, BSW]
	105	**Reserved** Format: MBZ
	104:85	**Source 1** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC [CHV, BSW]

madm - Multiply Add for Macro

	84	**Reserved** Format: MBZ
	83:64	**Source 0** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC [CHV, BSW]
	63:56	**Destination Register Number** Format: DstRegNum [CHV, BSW]
	55:53	**Destination Subregister Number** Format: DstSubRegNum[2:0]
	52:49	**Destination Channel Enable** Format: ChanEn[4] Four channel enables are defined for controlling which channels are written into the destination region. These channel mask bits are applied in a modulo-four manner to all ExecSize channels. There is 1-bit Channel Enable for each channel within the group of 4. If the bit is cleared, the write for the corresponding channel is disabled. If the bit is set, the write is enabled. Mnemonics for the bit being set for the group of 4 are x, y, z, and w, respectively, where x corresponds to Channel 0 in the group and w corresponds to channel 3 in the group
	48:42	**Reserved** Format: MBZ
	41:40	**Source 2 Modifier** Exists If: //([Property[Source Modifier]=='true') Format: SrcMod [CHV, BSW]
	39:38	**Source 1 Modifier** Exists If: //([Property[Source Modifier]=='true') Format: SrcMod [CHV, BSW]
	41:36	**Reserved** Exists If: //([Property[Source Modifier]=='false') Format: MBZ
	37:36	**Source 0 Modifier** Exists If: //([Property[Source Modifier]=='true') Format: SrcMod [CHV, BSW]
	35	**Reserved** Format: MBZ
	34	**Reserved** Format: MBZ
	33	**Flag Subregister Number** This field contains the flag subregister number for instructions with a non-zero Conditional Modifier.

madm - Multiply Add for Macro			
	32	**Reserved**	
		Format:	MBZ
	31:0	**Header**	
		Format:	EU_INSTRUCTION_HEADER [CHV, BSW]

No Operation

nop - No Operation

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

Do nothing. The nop instruction takes an instruction dispatch but performs no operation. It can be used for assembly patching in memory, or to insert a delay in the program sequence.

Format: nop

Restriction
The nop instruction takes no instruction options other than Breakpoint.

Syntax
nop

Pseudocode
{ ; // The null statement, which does nothing. }

Predication	Conditional Modifier	Saturation	Source Modifier
N	N	N	N

DWord	Bit	Description
0..3	127:31	**Reserved** Format: MBZ
	30	**Reserved**
	29:7	**Reserved** Format: MBZ
	6:0	**Opcode** Format: EU_OPCODE

Oword Block Read MSD

MSD0R_OWB - Oword Block Read MSD

Project:	CHV, BSW
Source:	DataPort 0
Length Bias:	1
Family:	Block R/W
Group:	OW Block R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR Indicates that the message requires a header.
	18	**Legacy Message** Default Value: 0h Project: All Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 00h Project: All Format: Opcode Oword Block Read message
	13	**Invalidate After Read** Project: All Format: MDC_IAR [CHV, BSW] Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12:11	**Reserved**
	10:8	**Data Elements** Project: All Format: MDC_DB_OW Specifies the number of contiguous Owords to be read or written
	7:0	**Binding Table Index** Project: All Format: MDC_BTS_A32 Specifies the Binding Table Index for the message

Oword Block Write MSD

MSD0W_OWB - Oword Block Write MSD

Project:	CHV, BSW
Source:	DataPort 0
Length Bias:	1
Family:	Block R/W
Group:	OW Block R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header.
	18	**Legacy Message** Default Value: 0h Project: All Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 08h Project: All Format: Opcode Oword Block Write message
	13:11	**Reserved** Project: All Format: MBZ Ignored
	10:8	**Data Elements** Project: All Format: MDC_DB_OW Specifies the number of contiguous Owords to be read or written
	7:0	**Binding Table Index** Project: All Format: MDC_BTS_A32 Specifies the Binding Table Index for the message

Oword Dual Block Read MSD

MSD0R_OWDB - Oword Dual Block Read MSD

Project:	CHV, BSW
Source:	DataPort 0
Length Bias:	1
Family:	Block R/W
Group:	OW Dual Block R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: Enable If set, indicates that the message includes the header.
	18	**Legacy Message** Default Value: 0h Project: All Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 02h Project: All Format: Opcode Oword Block Read message
	13	**Invalidate After Read** Project: All Format: MDC_IAR [CHV, BSW] Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12:10	**Reserved**
	9:8	**Data Elements** Project: All Format: MDC_DB_OWD [CHV, BSW] Specifies the number of contiguous Owords to be read or written
	7:0	**Binding Table Index** Project: All Format: MDC_BTS_A32 [CHV, BSW] Specifies the Binding Table Index for the message

Oword Dual Block Write MSD

MSD0W_OWDB - Oword Dual Block Write MSD

Project:	CHV, BSW
Source:	DataPort 0
Length Bias:	1
Family:	Block R/W
Group:	OW Dual Block R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: Enable If set, indicates that the message includes the header.
	18	**Legacy Message** Default Value: 0h Project: All Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 0Ah Project: All Format: Opcode Oword Block Read message
	13:10	**Reserved**
	9:8	**Data Elements** Project: All Format: MDC_DB_OWD [CHV, BSW] Specifies the number of contiguous Owords to be read or written
	7:0	**Binding Table Index** Project: All Format: MDC_BTS_A32 [CHV, BSW] Specifies the Binding Table Index for the message

Oword Unaligned Block Read MSD

MSD0R_OWUB - Oword Unaligned Block Read MSD

Project:	CHV, BSW
Source:	DataPort 0
Length Bias:	1
Family:	Block R/W
Group:	OW Unaligned Block R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header.
	18	**Legacy Message** Default Value: 0h Project: All Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 01h Project: All Format: Opcode Oword Unaligned Block Read message
	13:11	**Reserved**
	10:8	**Data Elements** Project: All Format: MDC_DB_OW [CHV, BSW] Specifies the number of contiguous Owords to be read
	7:0	**Binding Table Index** Project: All Format: MDC_BTS_A32 [CHV, BSW] Specifies the Binding Table Index for the message

PIPE_CONTROL

PIPE_CONTROL

Project:	CHV, BSW
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
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 4h DWORD_COUNT_n Project: CHV, BSW Format: =n Total Length - 2. Excludes DWord (0,1).
1	31:29	**Reserved** Project: All Format: MBZ
	28	**Reserved** Project: CHV, BSW Format: MBZ
	27	**Reserved** Project: CHV, BSW

PIPE_CONTROL

	26	**Reserved**
	25	**Reserved** Project: All Format: MBZ
	24	**Destination Address Type** Project: CHV, BSW Defines address space of Destination Address

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

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

	23	**LRI Post Sync Operation** Project: CHV, BSW

Value	Name	Description	Project
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.	All
1h	MMIO Write Immediate Data	Write the DWord contained in Immediate Data Low (DW3) to the MMIO offset specifed in the Address field.	All

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** Project: All
	21	**Store Data Index** Project: Format: U1 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** Project: All 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** One of the following must also be set: • Render Target Cache Flush Enable ([12] of DW1) • Depth Cache Flush Enable ([0] of DW1) • Stall at Pixel Scoreboard ([1] of DW1) • Depth Stall ([13] of DW1) • Post-Sync Operation ([13] of DW1) • DC Flush Enable([5] of DW1) This field must always be set in all PIPE_CONTROL commands having "Post-Sync Operation" set to "Report PS Depth Count" or "Report Time Stamp" when user wishes to set "Post-Sync Operation" bit in any of the MI_ATOMIC or MI_SEMAPHORE_SIGNAL commands programmed. This bit must be always set when PIPE_CONTROL command is programmed by GPGPU and MEDIA workloads, except for the cases when only Read Only Cache Invalidation bits are set (State Cache Invalidation Enable, Instruction cache Invalidation Enable, Texture Cache Invalidation Enable, Constant Cache Invalidation Enable). This is to WA FFDOP CG issue, this WA need not implemented when FF_DOP_CG is disable via "Fixed Function DOP Clock Gate Disable" bit in RC_PSMI_CTRL register.
	19	**Global Snapshot Count Reset** Project: All Format: U1 Value / Name / Description: 0h — Don't Reset — Do not reset the snapshot counts or Statistics Counters. 1h — Reset — Reset the snapshot count in Gen4 for all the units and reset the Statistics Counters except as noted above. **Programming Notes** This debug mode bit must not be exercised on any product. TIMESTAMP is not reset by PIPE_CONTROL with this bit set. When Post Sync Operation is set to "Write PS Depth Count" along with Global Snapshot Count Reset, PS Depth Count is Reported first before resetting the value.

PIPE_CONTROL

	Bits	Field
	18	**TLB Invalidate**
		Project: All
		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.
	17	**Reserved**
		Project: CHV, BSW
		Format: MBZ
	16	**Generic Media State Clear**
		Project: CHV, BSW
		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.
	15:14	**Post Sync Operation**
		Description
		This field specifies an optional action to be taken upon completion of the synchronization operation.
		This field must be cleared if the LRI Post-Sync Operation bit is set.

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

PIPE_CONTROL

		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**

Project:	All
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**

Project:	All
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.

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

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.

PIPE_CONTROL

11 — **Instruction Cache Invalidate Enable**

Project:	All
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**

Project:	All
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.

9 — **Indirect State Pointers Disable**

Project:	All
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.
[CHV, BSW]: 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**

Project:	All
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**

Project:	CHV, BSW
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**

PIPE_CONTROL

	5	**DC Flush Enable** Project: CHV, BSW 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** Project: All 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. **Workaround** / **Project** 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". — CHV, BSW
	3	**Constant Cache Invalidation Enable** Project: All 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** Project: All 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** Project: All Format: Enable Defines the behavior of PIPE_CONTROL command at the pixel scoreboard. Value / Name / Description / Project 0h / Disable / Stall at the pixel scoreboard is disabled. / All 1h / Enable / Stall at the pixel scoreboard is enabled. / All **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.

<table>
<tr><th colspan="3">PIPE_CONTROL</th></tr>
<tr><td></td><td>0</td><td>
Depth Cache Flush Enable
<table>
<tr><td>Project:</td><td>All</td></tr>
<tr><td>Format:</td><td>Enable</td></tr>
</table>
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.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th><th>Project</th></tr>
<tr><td>0h</td><td>Flush Disabled</td><td>Depth relates caches (HiZ, Stencil and Depth) are NOT flushed.</td><td>All</td></tr>
<tr><td>1h</td><td>Flush Enabled</td><td>Depth relates caches (HiZ, Stencil and Depth) are flushed.</td><td>All</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>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.</td></tr>
</table>
</td></tr>
<tr><td rowspan="2">2</td><td>31:2</td><td>
Address
<table>
<tr><td>Project:</td><td>CHV, BSW</td></tr>
<tr><td>Format:</td><td>GraphicsAddress[31:2]U32</td></tr>
</table>
If Post Sync Operation is set to 1h ([CHV, BSW]: 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 [CHV, BSW]: 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.
</td></tr>
<tr><td>1:0</td><td>Reserved</td></tr>
<tr><td rowspan="2">3
Project:
CHV, BSW</td><td>31:16</td><td>Reserved</td></tr>
<tr><td>15:0</td><td>
Address High
<table>
<tr><td>Project:</td><td>All</td></tr>
<tr><td>Format:</td><td>GraphicsAddress[47:32]U32</td></tr>
</table>
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.
</td></tr>
<tr><td>4..5
Project:
CHV, BSW</td><td>63:0</td><td>
Immediate Data
<table>
<tr><td>Project:</td><td>CHV, BSW</td></tr>
<tr><td>Format:</td><td>U64</td></tr>
</table>
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".
<table>
<tr><th>Programming Notes</th><th>Project</th></tr>
<tr><td>This field must be programmed to 0 when Post-Sync Operation is set to Write PS Depth Count or Write Timestamp.</td><td>CHV, BSW</td></tr>
</table>
</td></tr>
</table>

PIPELINE_SELECT

PIPELINE_SELECT

Project:	CHV, BSW
Source:	PRM
Length Bias:	1

Description
The PIPELINE_SELECT command is used to specify which GPE pipeline is to be considered the 'current' active pipeline. Issuing 3D-pipeline-specific commands when the Media pipeline is selected, or vice versa, is UNDEFINED.
Issuing 3D-pipeline-specific commands when the GPGPU pipeline is selected, or vice versa, is UNDEFINED.
Programming common non pipeline commands (e.g., STATE_BASE_ADDRESS) is allowed in all pipeline modes.

Programming Notes
Software must ensure all the write caches are flushed through a stalling PIPE_CONTROL command followed by another PIPE_CONTROL command to invalidate read only caches prior to programming MI_PIPELINE_SELECT command to change the Pipeline Select Mode. Example: ... Workload-3Dmode PIPE_CONTROL (CS Stall, Depth Cache Flush Enable, Render Target Cache Flush Enable, DC Flush Enable) PIPE_CONTROL (Constant Cache Invalidate, Texture Cache Invalidate, Instruction Cache Invalidate, State Cache invalidate) PIPELINE_SELECT (GPGPU)
Software must clear the **COLOR_CALC_STATE Valid** field in 3DSTATE_CC_STATE_POINTERS command prior to send a PIPELINE_SELECT with **Pipeline Select** set to GPGPU.
Render CS Only: SW must always program PIPE_CONTROL with CS Stall and Render Target Cache Flush Enable set prior to programming PIPELINE_SELECT command for GPGPU workloads i.e when pipeline mode is set to GPGPU. This is required to achieve better GPGPU preemption latencies for certain programming sequences. If programming PIPE_CONTROL has performance implications then preemption latencies can be trade off against performance by not implementing this programming note.
Hardware Binding Tables are only supported for 3D workloads. Resource streamer must be enabled only for 3D workloads. Resource streamer must be disabled for Media and GPGPU workloads. Batch buffer containing both 3D and GPGPU workloads must take care of disabling and enabling Resource Streamer appropriately while changing the PIPELINE_SELECT mode from 3D to GPGPU and vice versa. Resource streamer must be disabled using MI_RS_CONTROL command and Hardware Binding Tables must be disabled by programming 3DSTATE_BINDING_TABLE_POOL_ALLOC with "Binding Table Pool Enable" set to disable (i.e. value '0'). Example below shows disabling and enabling of resource streamer in a batch buffer for 3D and GPGPU workloads. MI_BATCH_BUFFER_START (Resource Streamer Enabled) PIPELINE_SELECT (3D) 3DSTATE_BINDING_TABLE_POOL_ALLOC (Binding Table Pool Enabled) 3D WORKLAOD MI_RS_CONTROL (Disable Resource Streamer) 3DSTATE_BINDING_TABLE_POOL_ALLOC (Binding Table Pool Disabled) PIPELINE_SELECT (GPGPU) GPGPU Workload 3DSTATE_BINDING_TABLE_POOL_ALLOC (Binding Table Pool Enabled) MI_RS_CONTROL (Enable Resource Streamer) 3D WORKLOAD MI_BATCH_BUFFER_END
Render command streamer (RCS) does Fixed Function DOP CG for media and gpgpu workloads. During context restore 3D semi pipeline state is restored to WM, which triggers implicit flush. Theoretically under certain conditions RCS can trigger FFDOP CG while WM in doing implicit flush leading to hangs. Only way to work around this issue is by disabling FFDOP CG feature. This issue is being fixed on G0 stepping.

PIPELINE_SELECT

So far this issue is not yet hit in pre-silicon validation, emulation, Silicon-SV or Silicon-Driver being a rare condition. Work around to disable FFDOP CG is not being applied at this point to allow valdiaiton to make progress on FFDOP CG feature, further decision will be taken when this issue occurs. Validation teams are requested to triage any GPGPU workload hangs with WM no done with FFDOP CG disabled to eliminate this scenario.

DWord	Bit	Description		
0	31:29	**Command Type**		
		Default Value:	3h GFXPIPE	
		Format:	OpCode	
	28:27	**Command SubType**		
		Default Value:	1h GFXPIPE_SINGLE_DW	
		Format:	OpCode	
	26:24	**3D Command Opcode**		
		Format:	OpCode	
		Value	**Name**	**Project**
		1h	GFXPIPE_NONPIPELINED **[Default]**	CHV, BSW
	23:16	**3D Command Sub Opcode**		
		Default Value:	04h GFXPIPE	
		Format:	OpCode	
	6	**Reserved**		
		Project:	CHV, BSW	
	15:2	**Reserved**		
		Project:	CHV, BSW	
	1:0	**Pipeline Selection**		
		Value	**Name**	**Description**
		0	3D	3D pipeline is selected
		1	Media	Media pipeline is selected (Includes HD optical disc playback, HD video playback, and generic media workloads)
		2	GPGPU	GPGPU pipeline is selected

Plane

pln - Plane

Project:	CHV, BSW
Source:	Eulsa
Length Bias:	4

The pln instruction computes a component-wise plane equation (w = p*u+q*v+r where u/v/w are vectors and p/q/r are scalars) of src0 and src1 and stores the results in dst. src1 is the input vector u. src0 provides input scalars p, q, and r, where p is the scalar value based on the region description of src0 and q and r are the scalar values implied from the src0 region. Specifically, q is the second component and r is the fourth component of the 4-tuple (128-bit aligned) that p belongs to.

Format: [(pred)] pln[.cmod] (exec_size) dst src0 src1

Restriction	Project
This is a specialized instruction that only supports an execution size (ExecSize) of 8 or 16.	
The src0 region must be a replicated scalar (with HorzStride == VertStride == 0).	
src0 must specify .0 or .4 as the subregister number, corresponding to a subregister byte offset of 0 or 16.	
Source operands cannot be accumulators.	CHV, BSW

Syntax
[(pred)] pln[.cmod] (exec_size) reg reg reg

Pseudocode

```
 Evaluate(WrEn);
for ( n = 0; n < exec_size; n++ ) {
    float dwP = src0.RegNum.SubRegNum[bits4:2];         // A DWord-aligned scalar.
    float dwQ = src0.RegNum.(SubRegNum[bit4:2] | 0x1);  // Second component.
    float dwR = src0.RegNum.(SubRegNum[bit4:2] | 0x3);  // Fourth component.
    if ( ExecSize == 8 ) {
        u = src1.RegNum
        v = src1.(RegNum + 1)
    } else {
        if ( n < 8 ) {
            u = src1.RegNum
            v = src1.(RegNum + 1)
        } else {
            u = src1.(RegNum + 2)
            v = src1.(RegNum + 3)
        }
    }

    if ( WrEn.chan[n] ) {
       dst.chan[n] = dwP * u.chan[n] + dwQ * v.chan[n] + dwR;
    }
}
```

pln - Plane

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	N

Src Types	Dst Types
F	F

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

REP16 Render Target Write MSD

MSD_RTW_REP16 - REP16 Render Target Write MSD

Project:	CHV, BSW
Source:	Render Cache DataPort
Length Bias:	1
Family:	Other
Group:	Render Target R/W

DWord	Bit	Description
0	31	**Reserved** Project: All Format: MBZ Ignored
	30	**Message Precision Subtype** Default Value: 0h Project: All Format: Opcode Full precision data message
	29	**Reserved** Project: All Format: MBZ Ignored
	28:25	**Message Length** Project: All Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Project: All Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the 2-register header.

MSD_RTW_REP16 - REP16 Render Target Write MSD

	18	**Reserved**
	17:14	**Message Type** Default Value: 0Ch Project: All Format: Opcode Render Target Write message
	13	**Reserved**
	12	**Last Render Target Select** Project: All Format: Enable This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero. **Programming Notes** When a pixel shader has render target writes at finer granularity than the dispatch rate, last render target write to a null surface must be present at the dispatch rate with this bit set. In particular, if a kernel is dispatched at pixel rate and it only writes to render targets at sample-rate, it must include a pixel-rate render target write to a null surface with Last Render Target Select bit enabled.
	11	**Slot Group Select** Project: All Format: MDC_RT_SGS [CHV, BSW] This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
	10:8	**Render Target Message Subtype** Default Value: 1h Project: All Format: Opcode SIMD16 Single source message with replicated data. Use slots [15:0] for pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [31:16] are referenced instead of [15:0].
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Return

ret - Return

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

Description
Return execution to the code sequence that called a subroutine. The ret instruction can be predicated or non-predicated. If non-predicated, all channels jump to the return IP in the first channel of src0 and restore CallMask from the second channel of src0. If predicated, the enabled channels jump to the return IP from the first channel of src0 and the corresponding bits in the CallMask are cleared to zero; if all CallMask bits are zero after the ret instruction, then execution jumps to the return IP from the first channel of src0. When SPF is on, the predication control must be scalar.
Format: [(pred)] ret (exec_size) null src0

Restriction
This instruction cannot take accumulator as source.
The src0 regioning control must be <2;2,1>

Syntax
[(pred)] ret (exec_size) null reg

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { PcIP[n] = src0.chan[0]; CallMask[n] = 0; } else { PcIP[n] = IP + 1; } } for (n = exec_size; n < 32; n++) { PcIP[n] = IP + 1; } if (CallMask[n:0] == 0)) { // all channels are zero Jump(src0.chan[0]); CallMask = src0.chan[1]; }

Predication	Conditional Modifier	Saturation	Source Modifier	Src Types
Y	N	N	N	D, UD

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32 [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Round Down

rndd - Round Down

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The rndd instruction takes component-wise floating point downward rounding (to the integral float number closer to negative infinity) of src0 and storing the rounded integral float results in dst. This is commonly referred to as the floor() function. Each result follows the rules in the following tables based on the floating-point mode.

Format: [(pred)] rndd[.cmod] (exec_size) dst src0

Syntax
[(pred)] rndd[.cmod] (exec_size) reg reg [(pred)] rndd[.cmod] (exec_size) reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = floor(src0.chan[n]); } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types
F	F

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32 [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Round to Nearest or Even

rnde - Round to Nearest or Even

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The rnde instruction takes component-wise floating point round-to-even operation of src0 with results in two pieces - a downward rounded integral float results stored in dst and the round-to-even increments stored in the rounding increment bits. The round-to-even increment must be added to the results in dst to create the final round-to-even values to emulate the round-to-even operation, commonly known as the round() function. The final results are the one of the two integral float values that is nearer to the input values. If the neither possibility is nearer, the even alternative is chosen. Each result follows the rules in the following tables based on the floating-point mode.

Format: [(pred)] rnde[.cmod] (exec_size) dst src0

Syntax
[(pred)] rnde[.cmod] (exec_size) reg reg [(pred)] rnde[.cmod] (exec_size) reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { if (src0.chan[n] - floor(src0.chan[n]) > 0.5f) { dst.chan[n] = floor(src0.chan[n]) + 1; } else if (src0.chan[n] - floor(src0.chan[n]) < 0.5f) { dst.chan[n] = floor(src0.chan[n]); } else { if (floor(src0.chan[n]) is odd) { dst.chan[n] = floor(src0.chan[n]) + 1; } else { dst.chan[n] = floor(src0.chan[n]); } } } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types
F	F

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32 [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Round to Zero

rndz - Round to Zero

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The rndz instruction takes component-wise floating point round-to-zero operation of src0 with results in two pieces - a downward rounded integral float results stored in dst and the round-to-zero increments stored in the rounding increment bits. The round-to-zero increment must be added to the results in dst to create the final round-to-zero values to emulate the round-to-zero operation, commonly known as the truncate() function. The final results are the one of the two closest integral float values to the input values that is nearer to zero.

Format: [(pred)] rndz[.cmod] (exec_size) dst src0

Syntax
[(pred)] rndz[.cmod] (exec_size) reg reg [(pred)] rndz[.cmod] (exec_size) reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = floor(src0.chan[n]); if (abs(src0.chan[n]) < abs(dst.chan[n])) { dst.chan[n] = floor(src0.chan[n]) + 1; } else { dst.chan[n] = floor(src0.chan[n]); } } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types
F	F

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32 [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Round Up

rndu - Round Up

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The rndu instruction takes component-wise floating point upward rounding (to the integral float number closer to positive infinity) of src0, commonly known as the ceiling() function. Each result follows the rules in the following tables based on the floating-point mode.

Format: [(pred)] rndu[.cmod] (exec_size) dst src0

Syntax
[(pred)] rndu[.cmod] (exec_size) reg reg [(pred)] rndu[.cmod] (exec_size) reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { if (src0.chan[n] - floor(src0.chan[n]) > 0.0f) { dst.chan[n] = floor(src0.chan[n]) + 1; } else { dst.chan[n] = src0.chan[n]; } } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types
F	F

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32 [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Scattered Move

smov - Scattered Move

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The smov instruction moves the components in src0 into dst. For each enabled channel, copy src0 to dst. The immediate is used to selectively enable channels without using flags. When predication is enabled, the predicate mask is not generated from the flags. Instead, the immediate is used to mask the execution mask. If any channel is enabled as a result of this masking, the instruction is executed. When predication is not enabled, the immediate masks the execution mask. This provides flexibility to mask out any channel with an immediate.

Format: [(pred)] smov[.cmod] (exec_size) dst src0 src1

Programming Notes
When predication is disabled, the immediate provides the flexibility to perform a select operation without the use of flags.
When predication is enabled, the usage model provides flexibility to select any bit in the flag registers for predication for execution size of 1.

Syntax
[(pred)] smov[.cmod] (exec_size) reg reg imm32

Pseudocode
if pred emask = OR (emask AND imm32) Else pmask = imm32. Evaluate(WrEn); for (n = 0; n < 32; n++) { if (WrEn.chan[n]) { dst.chan[n] = src0.chan[n]; } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Scratch Block Read MSD

MSD0R_HWB - Scratch Block Read MSD

Project:	CHV, BSW
Source:	DataPort 0
Length Bias:	1
Family:	Block R/W
Group:	HW Block R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header.
	18	**Scratch Block Message** Default Value: 1h Project: All Format: Opcode Scratch Block Message
	17	**Operation Type** Default Value: 0h Project: All Format: Opcode Scratch Block Read message
	16	**Channel Mode** Project: CHV, BSW Format: MDC_CMODE [CHV, BSW] Specifies whether the read or write operation occurs on all 4 Dwords if any of those channel enables are set, or else only on the dwords whose corresponding channel enable is set.
	15	**Invalidate After Read** Project: All Format: MDC_IAR [CHV, BSW] Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	14	**Reserved**

MSD0R_HWB - Scratch Block Read MSD

	13:12	**Data Elements** Project: All Format: MDC_DB_HW [CHV, BSW] Specifies the number of registers to be read or written
	11:0	**Address Offset** Project: All Format: GeneralStateOffset[17:6] HWORD (32 byte) based address offset to the BufferAddress in the Message Header.

Scratch Block Write MSD

MSD0W_HWB - Scratch Block Write MSD

Project:	CHV, BSW
Source:	DataPort 0
Length Bias:	1
Family:	Block R/W
Group:	HW Block R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header.
	18	**Scratch Block Message** Default Value: 1h Project: All Format: Opcode Scratch Block Message
	17	**Operation Type** Default Value: 1h Project: All Format: Opcode Scratch Block Write message
	16	**Channel Mode** Project: CHV, BSW Format: MDC_CMODE [CHV, BSW] Specifies whether the read or write operation occurs on all 4 Dwords if any of those channel enables are set, or else only on the dwords whose corresponding channel enable is set.
	15:14	**Reserved**
	13:12	**Data Elements** Project: All Format: MDC_DB_HW [CHV, BSW] Specifies the number of registers to be read or written
	11:0	**Address Offset** Project: All Format: GeneralStateOffset[17:6] HWORD (32 byte) based address offset to the BufferAddress in the Message Header.

Select

sel - Select

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

Description

The sel instruction selectively moves the components in src0 or src1 into the channels of dst based on the predication. On a channel by channel basis, if the channel condition is true, data in src0 is moved into dst. Otherwise, data in src1 is moved into dst.
As the predication is used to select the two sources, it is not included in the evaluation of WrEn. The predicate clause is mandatory if cmod is omitted/0000b. If both predication and the conditional modifier are omitted, the results are undefined.
If the conditional modifier is specified (not 0000b, a compare is performed and the resulting condition flag is used for the sel instruction. Conditional modifiers .ge and .l follow the cmpn rules, and all other conditional modifiers follow the cmp rules. Predication is not allowed in this mode.
A sel instruction with cmod .l is used to emulate a MIN instruction.
A sel instruction with cmod .ge is used to emulate a MAX instruction.
For a sel instruction with a .l or .ge conditional modifier, if one source is NaN and the other not NaN, the non-NaN source is the result. If both sources are NaNs, the result is NaN. For all other conditional modifiers, if either source is NaN then src1 is selected.
A sel instruction without a conditional modifier always copies a denorm source value to a denorm destination value (in the manner of a raw move). This applies even if the source modifies are set on the *sel* instruction sources.
The sel instruction uses any conditional modifier internally and does not update the flag register if a conditional modifier is used.

A sel instruction with cmod or source modifier will flush denorm to zero, depending on the denorm mode bit; a sel instruction without cmod and source modifier will retain denorm.

Format: (pred) sel[.cmod] (exec_size) dst src0 src1

Restriction

Predicated sel instruction cannot be used in mixed mode operation with packed half float destination.

Syntax

(pred) sel[.cmod] (exec_size) reg reg reg (pred) sel[.cmod] (exec_size) reg reg imm32

Pseudocode

```
Evaluate(WrEn, NoPMask); if (cmod == "0000") { // no CMod Evaluate(PMask); for ( n = 0; n < exec_size; n++ ) { if ( WrEn.chan[n] ) { if ( PMask.channel[n] ) { dst.chan[n] = src0.chan[n]; } else { dst.chan[n] = src1.chan[n]; } } } } else { // with CMod Evaluate(CMod); for ( n = 0; n < exec_size; n++ ) { if ( WrEn.chan[n] ) { if ( CMod.chan[n] ) { dst.chan[n] = src0.chan[n]; } else { dst.chan[n] = src1.chan[n]; } } } }
```

sel - Select

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types	Project
*B,*W*D	*B,*W,*D	
F	F	
DF	DF	CHV, BSW
*W,*D,*Q	*W,*D,*Q	CHV, BSW
HF	HF	CHV, BSW
HF, F	HF, F	CHV, BSW

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Shift Left

shl - Shift Left

Project:	CHV, BSW
Source:	Eulsa
Length Bias:	4

Description
Perform component-wise logical left shift of the bits in src0 by the shift count indicated in src1, storing the results in dst, inserting zero bits in the number of LSBs indicated by the shift count. Hardware detects overflow properly and uses it to perform any saturation operation on the result, as long as the shifted result is within 33 bits. Otherwise, the result is undefined. Note: For word and DWord operands, the accumulators have 33 bits.
In QWord mode, the shift count is taken from the low six bits of src1 regardless of the src1 type and treated as an unsigned integer in the range 0 to 63. Otherwise the shift count is taken from the low five bits of src1 regardless of the src1 type and treated as an unsigned integer in the range 0 to 31. The operation uses QWord mode if src0 or dst has the Q or UQ type but not if src1 is the only operand with the Q or UQ type.
Format: [(pred)] shl[.cmod] (exec_size) dst src0 src1

Restriction
Accumulator cannot be destination, implicit or explicit.

Syntax
[(pred)] shl[.cmod] (exec_size) reg reg reg [(pred)] shl[.cmod] (exec_size) reg reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { shiftCnt = src0 or dst has Q or UQ type ? src1.chan[n] & 0x3F : src1.chan[n] & 0x1F dst.chan[n] = src0.chan[n] « shiftCnt; } }

Predication	Conditional Modifier	Saturation	Source Modifier	Src Types	Dst Types
Y	Y	Y	Y	*B,*W,*D	*B,*W,*D

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Shift Right

shr - Shift Right

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

Description
Perform component-wise logical right shift with zero insertion of the bits in src0 by the shift count indicated in src1, storing the results in dst. Insert zero bits in the number of MSBs indicated by the shift count. src0 and dst can have different types and can be signed or unsigned. Note: For word and DWord operands, the accumulators have 33 bits. Note: For unsigned src0 types, shr and asr produce the same result.
In QWord mode, the shift count is taken from the low six bits of src1 regardless of the src1 type and treated as an unsigned integer in the range 0 to 63. Otherwise the shift count is taken from the low five bits of src1 regardless of the src1 type and treated as an unsigned integer in the range 0 to 31. The operation uses QWord mode if src0 or dst has the Q or UQ type but not if src1 is the only operand with the Q or UQ type.
Format: [(pred)] shr[.cmod] (exec_size) dst src0 src1

Syntax
[(pred)] shr[.cmod] (exec_size) reg reg reg [(pred)] shr[.cmod] (exec_size) reg reg imm32

Pseudocode	Project
Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { shiftCnt = src0 or dst has Q or UQ type ? src1.chan[n] & 0x3F : src1.chan[n] & 0x1F dst.chan[n] = src0.chan[n] » shiftCnt; } }	CHV, BSW

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types	Project
UB, UW, UD	UB, UW, UD	
UW, UD, UQ	UW, UD, UQ	CHV, BSW

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

SIMD8 Render Target Write MSD

MSD_RTW_SIMD8 - SIMD8 Render Target Write MSD

Project:	CHV, BSW
Source:	Render Cache DataPort
Length Bias:	1
Family:	Other
Group:	Render Target R/W

DWord	Bit	Description
0	31	**Reserved**
	30	**Message Precision Subtype** Default Value: 0h Project: All Format: Opcode Full precision data message
	29	**Reserved**
	28:25	**Message Length** Project: All Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Project: All Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the 2-register header.
	18	**Reserved**
	17:14	**Message Type** Default Value: 0Ch Project: All Format: Opcode Render Target Write message

MSD_RTW_SIMD8 - SIMD8 Render Target Write MSD

	13	**Reserved** Project: CHV, BSW Format: MBZ Ignored
	12	**Last Render Target Select** Project: All Format: Enable This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero. **Programming Notes** When a pixel shader has render target writes at finer granularity than the dispatch rate, last render target write to a null surface must be present at the dispatch rate with this bit set. In particular, if a kernel is dispatched at pixel rate and it only writes to render targets at sample-rate, it must include a pixel-rate render target write to a null surface with Last Render Target Select bit enabled.
	11	**Slot Group Select** Project: All Format: MDC_RT_SGS [CHV, BSW] This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
	10:8	**Render Target Message Subtype** Default Value: 4h Project: All Format: Opcode SIMD8 single source message. Use slots [7:0] for pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [23:16] are referenced instead of [7:0].
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

SIMD16 Render Target Write MSD

MSD_RTW_SIMD16 - SIMD16 Render Target Write MSD

Project:	CHV, BSW
Source:	Render Cache DataPort
Length Bias:	1
Family:	Other
Group:	Render Target R/W

DWord	Bit	Description
0	31	**Reserved**
	30	**Message Precision Subtype** Default Value: 0h Project: All Format: Opcode Full precision data message
	29	**Reserved**
	28:25	**Message Length** Project: All Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Project: All Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the 2-register header.
	18	**Reserved**
	17:14	**Message Type** Default Value: 0Ch Project: All Format: Opcode Render Target Write message

MSD_RTW_SIMD16 - SIMD16 Render Target Write MSD

	13	**Reserved** Project: CHV, BSW Format: MBZ Ignored
	12	**Last Render Target Select** Project: All Format: Enable This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero. **Programming Notes** When a pixel shader has render target writes at finer granularity than the dispatch rate, last render target write to a null surface must be present at the dispatch rate with this bit set. In particular, if a kernel is dispatched at pixel rate and it only writes to render targets at sample-rate, it must include a pixel-rate render target write to a null surface with Last Render Target Select bit enabled.
	11	**Slot Group Select** Project: All Format: MDC_RT_SGS [CHV, BSW] This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
	10:8	**Render Target Message Subtype** Default Value: 0h Project: All Format: Opcode SIMD16 Single source message. Use slots [15:0] for pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [31:16] are referenced instead of [15:0].
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

STATE_BASE_ADDRESS

STATE_BASE_ADDRESS

Project:	CHV, BSW
Source:	PRM
Length Bias:	2

The STATE_BASE_ADDRESS command sets the base pointers for subsequent state, instruction, and media indirect object accesses by the GPE.

For more information see the Base Address Utilization table in the Memory Access Indirection narrative topic.

Programming Notes	Project
The following commands must be reissued following any change to the base addresses: • 3DSTATE_CC_POINTERS • 3DSTATE_BINDING_TABLE_POINTERS • 3DSTATE_SAMPLER_STATE_POINTERS • 3DSTATE_VIEWPORT_STATE_POINTERS • MEDIA_STATE_POINTERS Execution of this command causes a full pipeline flush, thus its use should be minimized for higher performance.	
SW must always program PIPE_CONTROL with "CS Stall" and "Render Target Cache Flush Enable" set before programming STATE_BASE_ADDRESS command for GPGPU workloads i.e when pipeline select is GPGPU via PIPELINE_SELECT command. This is required to achieve better GPGPU preemption latencies in certain workload programming sequences. If programming PIPE_CONTROL has performance implications then preemption latencies can be traded off against performance by not implementing this programming note.	CHV, BSW

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE
	28:27	**Command SubType** Default Value: 0h GFXPIPE_COMMON
	26:24	**3D Command Opcode** Default Value: 1h GFXPIPE_NONPIPELINED
	23:16	**3D Command Sub Opcode** Default Value: 01h STATE_BASE_ADDRESS
	15:8	**Reserved** Project: All Format: MBZ

STATE_BASE_ADDRESS

DWord	Bit	Description
	7:0	**DWord Length**

Project:	CHV, BSW
Format:	=n Total Length - 2

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

DWord	Bit	Description
1..2	63:12	**General State Base Address**

Project:	All
Format:	GraphicsAddress[63:12]

Specifies the 4K-byte aligned base address for general state accesses. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].

Programming Notes
Bounds checking is performed on general state accesses by Data Port Shared Functions for stateless A32 messages. Bounds checking is enabled when General State Base Address [46:12] + General State Buffer Size [31:12] is <= 2^47. This ensures that the General State Buffer does not straddle the canonical address boundary where GraphicsAddress [47] changes.

Restriction
General State Base Address [47:12] + General State Buffer Size [31:12] must be < 2^48. It is illegal programming for this to be >= 2^48.
When using stateless (A32) Data Port messages, General State Base Address [47:12] + Buffer Base Address [31:0] must be < 2^48. It is illegal for this to be >= 2^48.

DWord	Bit	Description
	11	**Reserved**

Project:	All
Format:	MBZ

DWord	Bit	Description
	10:4	**General State Memory Object Control State**

Project:	All
Format:	MEMORY_OBJECT_CONTROL_STATE

Specifies the memory object control state for indirect state using the **General State Base Address**, with the exception of the stateless data port accesses.

DWord	Bit	Description
	3:1	**Reserved**
	0	**General State Base Address Modify Enable**

Project:	All
Format:	Enable

The other fields in this DWord and the following DWord are updated only when this bit is set.

Value	Name	Description	Project
0h	Disable	Ignore the updated address.	All
1h	Enable	Modify the address.	All

STATE_BASE_ADDRESS

DWord	Bit	Description
3	31:23	**Reserved** Project: All Format: MBZ
	22:16	**Stateless Data Port Access Memory Object Control State** Project: All Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for stateless data port accesses.
	15:0	**Reserved** Project: All Format: MBZ
4..5	63:12	**Surface State Base Address** Project: All Format: GraphicsAddress[63:12] Specifies the 4K-byte aligned base address for binding table and surface state accesses. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].
	11	**Reserved** Project: All Format: MBZ
	10:4	**Surface State Memory Object Control State** Project: All Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for indirect state using the **Surface State Base Address**.
	3:1	**Reserved** Project: All Format: MBZ
	0	**Surface State Base Address Modify Enable** Project: All Format: Enable The other fields in this DWord and the following DWord are updated only when this bit is set.

Value	Name	Description	Project
0h	Disable	Ignore the updated address.	All
1h	Enable	Modify the address.	All

STATE_BASE_ADDRESS

DWord	Bit	Description
		Programming Notes Setting this bit to 1 in a batch buffer causes the resource streamer to stop; for performance reasons the SW should only place commands with this bit set in the ring buffer. Before programming the Surface State Base Address, the RS must be disabled. Within a batch buffer where the RS is enabled, RS may be disabled thru a MI_RS_CONTROL command with Resource Streamer Control cleared prior to the STATE_BASE_ADDRESS with Surface State Base Address Modify Enable set and then re-enabled with another MI_RS_CONTROL with Resource Streamer Control set.
6..7	63:12	**Dynamic State Base Address** Project: All Format: GraphicsAddress[63:12] Specifies the 4K-byte aligned base address for sampler and viewport state accesses. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].
	11	**Reserved** Project: All Format: MBZ
	10:4	**Dynamic State Memory Object Control State** Project: All Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for indirect state using the **Dynamic State Base Address**. Push constants defined in 3DSTATE_CONSTANT_(VS | GS | PS) commands do not use this control state, although they can use the corresponding base address. The memory object control state for push constants is defined within the command.
	3:1	**Reserved** Project: All Format: MBZ
	0	**Dynamic State Base Address Modify Enable** Project: All Format: Enable The other fields in this DWord and the following DWord are updated only when this bit is set. Value | Name | Description | Project 0h | Disable | Ignore the updated address. | All 1h | Enable | Modify the address. | All
8..9	63:12	**Indirect Object Base Address** Project: All Format: GraphicsAddress[63:12]IndirectObject Specifies the 4K-byte aligned base address for indirect object load in MEDIA_OBJECT command.

<table>
<tr><th colspan="3">STATE_BASE_ADDRESS</th></tr>
<tr><td></td><td>11</td><td>Reserved
Project: All
Format: MBZ</td></tr>
<tr><td></td><td>10:4</td><td>Indirect Object Memory Object Control State
Project: All
Format: MEMORY_OBJECT_CONTROL_STATE
Specifies the memory object control state for indirect objects using the Indirect Object Base Address.</td></tr>
<tr><td></td><td>3:1</td><td>Reserved
Project: All
Format: MBZ</td></tr>
<tr><td></td><td>0</td><td>Indirect Object Base Address Modify Enable
Project: All
Format: Enable
The other fields in this DWord and the following DWord are updated only when this bit is set.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th><th>Project</th></tr>
<tr><td>0h</td><td>Disable</td><td>Ignore the updated address.</td><td>All</td></tr>
<tr><td>1h</td><td>Enable</td><td>Modify the address.</td><td>All</td></tr>
</table></td></tr>
<tr><td>10..11</td><td>63:12</td><td>Instruction Base Address
Project: All
Format: GraphicsAddress[63:12]
Specifies the 4K-byte aligned base address for all EU instruction accesses.
GraphicsAddress[63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].</td></tr>
<tr><td></td><td>11</td><td>Reserved
Project: All
Format: MBZ</td></tr>
<tr><td></td><td>10:4</td><td>Instruction Memory Object Control State
Project: All
Format: MEMORY_OBJECT_CONTROL_STATE
Specifies the memory object control state for EU instructions using the Instruction Base Address.</td></tr>
<tr><td></td><td>3:1</td><td>Reserved
Project: All
Format: MBZ</td></tr>
</table>

STATE_BASE_ADDRESS

DWord	Bit	Description
	0	**Instruction Base Address Modify Enable** Project: All Format: Enable The other fields in this DWord and the following DWord are updated only when this bit is set.

Value	Name	Description	Project
0h	Disable	Ignore the updated address.	All
1h	Enable	Modify the address.	All

DWord	Bit	Description
12	31:12	**General State Buffer Size** Project: All Format: U20

FormatDesc
This field specifies the size of the buffer in 4K pages. Any access that straddles or goes past the end of the buffer returns 0. Note that BufferSize=0 indicates that there is no valid data in the buffer.

Workaround	Project
When the General State Buffer Size is programmed to 0, SLM accesses are treated as out-of-bounds (should only apply to Stateless accesses). Workaround is to program the General State Buffer Size to a value > 0.	CHV, BSW
If Per Thread Scratch Space Size bounds checking is enabled by GT_MODE[15], then General State Buffer Size must be set larger than the maximum Per Thread Scratch Space Size.	CHV, BSW [CHV, BSW]

DWord	Bit	Description
	11:1	**Reserved**
	0	**General State Buffer Size Modify Enable** Project: All Format: Enable The bound in this DWord is updated only when this bit is set.

Value	Name	Description	Project
0h	Disable	Ignore the updated bound.	All
1h	Enable	Modify the updated bound.	All

DWord	Bit	Description
13	31:12	**Dynamic State Buffer Size** Project: All Format: U20

FormatDesc
This field specifies the size of the buffer in 4K pages. Any access that straddles or goes past the end of the buffer returns 0. Note that BufferSize=0 indicates that there is no valid data in the buffer.

STATE_BASE_ADDRESS

DWord	Bit	Description
	11:1	**Reserved**
	0	**Dynamic State Buffer Size Modify Enable** Project: All Format: Enable FormatDesc The bound in this DWord is updated only when this bit is set. Value / Name / Description / Project: 0h / Disable / Ignore the updated bound. / All 1h / Enable / Modify the updated bound. / All
14	31:12	**Indirect Object Buffer Size** Project: All Format: U20 FormatDesc This field specifies the size of the buffer in 4K pages. Any access that straddles or goes past the end of the buffer returns 0. Note that BufferSize=0 indicates that there is no valid data in the buffer.
	11:1	**Reserved**
	0	**Indirect Object Buffer Size Modify Enable** Project: All Format: Enable FormatDesc The bound in this DWord is updated only when this bit is set. Value / Name / Description / Project: 0h / Disable / Ignore the updated bound. / All 1h / Enable / Modify the updated bound. / All
15	31:12	**Instruction Buffer Size** Project: All Format: U20 FormatDesc This field specifies the size of the buffer in 4K pages. Any access that straddles or goes past the end of the buffer returns 0. Note that BufferSize=0 indicates that there is no valid data in the buffer.
	11:1	**Reserved**

STATE_BASE_ADDRESS		
	0	**Instruction Buffer size Modify Enable**

Project:	All
Format:	Enable

FormatDesc
The bound in this DWord is updated only when this bit is set.

Value	Name	Description	Project
0h	Disable	Ignore the updated bound.	All

STATE_PREFETCH

STATE_PREFETCH

Project:	CHV, BSW
Source:	PRM
Length Bias:	2

(This command is provided strictly for performance optimization opportunities, and likely requires some experimentation to evaluate the overall impact of additional prefetching.)
The STATE_PREFETCH command causes the GPE to attempt to prefetch a sequence of 64-byte cache lines into the GPE-internal cache ("L2 ISC") used to access EU kernel instructions and fixed/shared function indirect state data. While state descriptors, surface state, and sampler state are automatically prefetched by the GPE, this command may be used to prefetch data not automatically prefetched, such as: 3D viewport state; Media pipeline Interface Descriptors; EU kernel instructions.

Restriction	Project
Restriction: Due to know HW issue this command doesn't achieve its intended purpose and must not be exercised/programmed by SW.	CHV, BSW

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE
	28:27	**Command SubType** Default Value: 0h GFXPIPE_COMMON
	26:24	**3D Command Opcode** Default Value: 0h GFXPIPE_PIPELINED
	23:16	**3D Command Sub Opcode** Default Value: 03h STATE_PREFETCH
	15:8	**Reserved**
	7:0	**DWord Length** Project: All Format: =n Total Length - 2 Value / Name / Description: 0h / DWORD_COUNT_n **[Default]** / Excludes DWord (0,1)
1	31:6	**Prefetch Pointer** Project: All Format: GraphicsAddress[31:6] Specifies the 64-byte aligned address to start the prefetch from. This pointer is an absolute virtual address, it is not relative to any base pointer.
	5:3	**Reserved**

STATE_PREFETCH		
	2:0	**Prefetch Count** Project: All Format: U3-1 count of cache lines Indicates the number of contiguous 64-byte cache lines that will be prefetched. Value: [0,7] — Name: — Description: indicating a count of [1,8]

STATE_SIP

STATE_SIP

Project:	All
Source:	PRM
Length Bias:	2

The STATE_SIP command specifies the starting instruction location of the System Routine that is shared by all threads in execution.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE
	28:27	**Command SubType** Default Value: 0h GFXPIPE_COMMON
	26:24	**3D Command Opcode** Default Value: 1h GFXPIPE_NONPIPELINED
	23:16	**3D Command Sub Opcode** Default Value: 02h STATE_SIP
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Project: All Format: =n Total Length - 2 Value: 1h; Name: DWORD_COUNT_n **[Default]**; Description: Excludes DWord (0,1); Project: CHV, BSW
1..2 **Project:** CHV, BSW	63:4	**System Instruction Pointer** Project: All Format: InstructionBaseOffset[63:4]Kernel Specifies the instruction address of the system routine associated with the current context as a 128-bit granular offset from the Instruction Base Address. SIP is shared by all threads in execution. The address specifies the double quadword aligned instruction location. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47]. **Programming Notes** This portion of the command is not context save/restored. The context image may restore this command as a 2 dword command rather than a 3 dword command.
	3:0	**Reserved**

Sum of Absolute Difference 2

sad2 - Sum of Absolute Difference 2

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The sad2 instruction takes source data channels from src0 and src1 in groups of 2-tuples. For each 2-tuple, it computes the sum-of-absolute-difference (SAD) between src0 and src1 and stores the scalar result in the first channel of the 2-tuple in dst. The results are also stored in the accumulator register. The destination operand and the accumulator maintain 16 bits per channel precision. The destination register must be aligned to even word (DWord). The even words in the destination region will contain the correct data. The odd words are also written but with undefined values.

Format: [(pred)] sad2[.cmod] (exec_size) dst src0 src1

Restriction
Source operands cannot be accumulators.
The execution size cannot be 1 as the computation requires at least two data channels.

Syntax
[(pred)] sad2[.cmod] (exec_size) reg reg reg [(pred)] sad2[.cmod] (exec_size) reg reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n += 2) { if (WrEn.chan[n]) { dst.chan[n] = abs(src0.chan[n] - src1.chan[n]) + abs(src0.chan[n+1] - src1.chan[n+1]); } }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types
B, UB	W, UW

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

Sum of Absolute Difference Accumulate 2

sada2 - Sum of Absolute Difference Accumulate 2

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

The sada2 instruction takes source data channels from src0 and src1 in groups of 2-tuples. For each 2-tuple, it computes the sum-of-absolute-difference (SAD) between src0 and src1, adds the intermediate result with the accumulator value corresponding to the first channel, and stores the scalar result in the first channel of the 2-tuple in dst. The destination operand and the accumulator maintain 16 bits per channel precision. Higher precision (guide bits) stored in the accumulator allows up to 64 rounds of sada2 instructions to be issued back to back without overflowing the accumulator. The destination register must be aligned to even word (DWord). The even words in the destination region will contain the correct data. The odd words are also written but with undefined values.

Format: [(pred)] sada2[.cmod] (exec_size) dst src0 src1

Restriction
Source operands cannot be accumulators.
The execution size cannot be 1 as the computation requires at least two data channels.

Syntax
[(pred)] sada2[.cmod] (exec_size) reg reg reg [(pred)] sada2[.cmod] (exec_size) reg reg imm32

Pseudocode
Evaluate(WrEn); for (n = 0; n < exec_size; n += 2) { uwTmp = abs(src0.chan[n] - src1.chan[n]) + abs(src0.chan[n+1] - src1.chan[n+1]); if (WrEn.chan[n]) { dst.chan[n] = uwTmp + acc[n]; } }

Predication	Conditional Modifier	Saturation	Source Modifier	Src Types	Dst Types
Y	Y	Y	Y	B, UB	W, UW

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG [CHV, BSW]
	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM [CHV, BSW]
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

SWTESS_BASE_ADDRESS

SWTESS_BASE_ADDRESS	
Project:	CHV, BSW
Source:	PRM
Length Bias:	2

The SWTESS_BASE_ADDRESS command sets the base pointers for SW Tessellation data read access by the TE unit.

Programming Notes
This base address must also be comprehended in the SURFACE_STATE used by the HS kernel to write the SW tessellation data. Execution of this command causes a full pipeline flush, thus its use should be minimized for higher performance.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE
	28:27	**Command SubType** Default Value: 0h GFXPIPE_COMMON
	26:24	**3D Command Opcode** Default Value: 1h GFXPIPE_NONPIPELINED
	23:16	**3D Command Sub Opcode** Default Value: 03h SWTESS_BASE_ADDRESS
	15:8	**Reserved**
	7:0	**DWord Length** Project: All Format: =n Total Length - 2 Value: 0h; Name: DWORD_COUNT_n **[Default]**; Description: Excludes DWord (0,1)
1	31:12	**SW Tessellation Base Address** Project: All Format: GraphicsAddress[31:12] Specifies the 4K-byte aligned base address for TE unit SW tessellation data read accesses.
	11:8	**SW Tessellation Memory Object Control State** Project: All Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state used by the TE unit to read SW tessellation data from memory.

SWTESS_BASE_ADDRESS		
	7:0	**Reserved** Project: All Format: MBZ
2 **Project:** CHV, BSW	31:16	**Reserved** Project: All Format: MBZ
	15:0	**SW Tessellation Base Address High** Project: CHV, BSW Format: GraphicsAddress[47:32] Specifies most significant bits of the 4K-byte aligned base address for TE unit SW tessellation data read accesses. See SW Tessellation Base Address[31:12] in DWord 0.

Typed Surface Read MSD

MSD1R_TS - Typed Surface Read MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Typed Surface R/W
Group:	Scattered Typed Surface R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 05h Project: All Format: Opcode Typed Surface Read message
	13:12	**Slot Group** Project: All Format: MDC_SG3 [CHV, BSW] Specifies the Slot Group mode of the message (which slots are processed)
	11:8	**Channel Mask** Project: All Format: MDC_CMASK [CHV, BSW] Specifies which RGBA channels are included in the message payload.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Typed Surface Write MSD

MSD1W_TS - Typed Surface Write MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Typed Surface R/W
Group:	Scattered Typed Surface R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 0Dh Project: All Format: Opcode Typed Surface Write message
	13:12	**Slot Group** Project: All Format: MDC_SG3 [CHV, BSW] Specifies the Slot Group mode of the message (which slots are processed)
	11:8	**Channel Mask** Project: All Format: MDC_CMASK [CHV, BSW] Specifies which RGBA channels are included in the message payload.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS [CHV, BSW] Specifies the Binding Table Index for the message

Untyped Surface Read MSD

MSD1R_US - Untyped Surface Read MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Surface R/W
Group:	Scattered Untyped Surface R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 01h Project: All Format: Opcode Untyped Surface Read message
	13:12	**SIMD Mode** Project: All Format: MDC_SM3 [CHV, BSW] Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Channel Mask** Project: All Format: MDC_CMASK [CHV, BSW] Specifies which RGBA channels are included in the message payload.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS_SLM_A32 [CHV, BSW] Specifies the Binding Table Index for the message

Untyped Surface Write MSD

MSD1W_US - Untyped Surface Write MSD

Project:	CHV, BSW
Source:	DataPort 1
Length Bias:	1
Family:	Untyped Surface R/W
Group:	Scattered Untyped Surface R/W

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHP [CHV, BSW] If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 09h Project: All Format: Opcode Untyped Surface Write message
	13:12	**SIMD Mode** Project: All Format: MDC_SM3 [CHV, BSW] Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Channel Mask** Project: All Format: MDC_UW_CMASK [CHV, BSW] Specifies which RGBA channels are included in the message payload.
	7:0	**Binding Table Index** Project: All Format: MDC_BTS_SLM_A32 [CHV, BSW] Specifies the Binding Table Index for the message

URB Hword Dual Block Read MSD

MSD_UR_HWDB - URB Hword Dual Block Read MSD

Project:	CHV, BSW
Source:	Read-Only DataPort
Length Bias:	1

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header.
	18	**Reserved**
	17	**Per Slot Offset** Format: MHC_PSOP Specifies if per-slot offsets are present and will be added to the **Global Offset**.
	16	**Reserved**
	15	**Swizzle Control** Project: All Format: Opcode Value: 1 — Name: URB_INTERLEAVED **[Default]** — Description: Use two URB entries (**URB Handle 0** and **URB Handle 1**). — Project: All
	14:4	**Global Offset** Project: All Format: U11 Specifes the offset, in units of Hword elements, from the start of the URB handle for the access. If **Per Slot Offset** is set, the global offset is added to those offsets to form the overall offset. Range [0,1023]
	3:0	**URB Opcode** Project: All Format: Opcode Value: 2 — Name: URB_READ_HWORD **[Default]** — Description: URB Hword Read message — Project: All

URB Hword Dual Block Write MSD

MSD_UW_HWDB - URB Hword Dual Block Write MSD

Project: CHV, BSW
Source: Read-Only DataPort
Length Bias: 1

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header.
	18	**Reserved**
	17	**Per Slot Offset** Format: MHC_PSOP Specifies if per-slot offsets are present and will be added to the **Global Offset**.
	16	**Reserved**
	15	**Swizzle Control** Project: All Format: Opcode Value: 1 — Name: URB_INTERLEAVED **[Default]** — Description: Use two URB entries (**URB Handle 0** and **URB Handle 1**). — Project: All
	14:4	**Global Offset** Project: All Format: U11 Specifes the offset, in units of Hword elements, from the start of the URB handle for the access. If **Per Slot Offset** is set, the global offset is added to those offsets to form the overall offset. Range [0,1023]
	3:0	**URB Opcode** Project: All Format: Opcode Value: 0 — Name: URB_WRITE_HWORD **[Default]** — Description: URB Hword Read message — Project: All

URB Oword Block Write MSD

MSD_UW_OWB - URB Oword Block Write MSD

Project: CHV, BSW
Source: Read-Only DataPort
Length Bias: 1

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header.
	18	**Reserved**
	17	**Per Slot Offset** Format: MHC_PSOP Specifies if per-slot offsets are present and will be added to the **Global Offset**.
	16	**Reserved**
	15	**Swizzle Control** Project: All Format: Opcode Value: 0 / Name: URB_NOSWIZZLE **[Default]** / Description: Use a single URB entry (**URB Handle 0**). / Project: All
	14:4	**Global Offset** Project: All Format: U11 Specifes the offset, in units of Oword elements, from the start of the URB handle for the access. If **Per Slot Offset** is set, the global offset is added to those offsets to form the overall offset. Range [0,2047]
	3:0	**URB Opcode** Project: All Format: Opcode Value: 1 / Name: URB_WRITE_OWORD **[Default]** / Description: URB Oword Write message / Project: All

URB Oword Dual Block Read MSD

MSD_UR_OWDB - URB Oword Dual Block Read MSD

Project:	CHV, BSW
Source:	Read-Only DataPort
Length Bias:	1

DWord	Bit	Description
0	19	**Header Present** Project: All Format: MDC_MHR [CHV, BSW] Indicates that the message requires a header.
	18	**Reserved**
	17	**Per Slot Offset** Format: MHC_PSOP Specifies if per-slot offsets are present and will be added to the **Global Offset**.
	16	**Reserved**
	15	**Swizzle Control** Project: All Format: Opcode Value: 1 | Name: URB_INTERLEAVED **[Default]** | Description: Use two URB entries (**URB Handle 0** and **URB Handle 1**). | Project: All
	14:4	**Global Offset** Project: All Format: U11 Specifes the offset, in units of Oword elements, from the start of the URB handle for the access. If **Per Slot Offset** is set, the global offset is added to those offsets to form the overall offset. Range [0,2047]
	3:0	**URB Opcode** Project: All Format: Opcode Value: 3 | Name: URB_READ_OWORD **[Default]** | Description: URB Oword Read message | Project: All

URB Oword Dual Block Write MSD

MSD_UW_OWDB - URB Oword Dual Block Write MSD

Project: CHV, BSW

Source: Read-Only DataPort

Length Bias: 1

DWord	Bit	Description
0	19	**Header Present**
		Project: All
		Format: MDC_MHR [CHV, BSW]
		Indicates that the message requires a header.
	18	**Reserved**
	17	**Per Slot Offset**
		Format: MHC_PSOP
		Specifies if per-slot offsets are present and will be added to the **Global Offset**.
	16	**Reserved**
	15	**Swizzle Control**
		Project: All
		Format: Opcode
	14:4	**Global Offset**
		Project: All
		Format: U11
		Specifes the offset, in units of Oword elements, from the start of the URB handle for the access. If **Per Slot Offset** is set, the global offset is added to those offsets to form the overall offset. Range [0,2047]
	3:0	**URB Opcode**
		Project: All
		Format: Opcode

Swizzle Control values:

Value	Name	Description	Project
1	URB_INTERLEAVED **[Default]**	Use two URB entries (**URB Handle 0** and **URB Handle 1**).	All

URB Opcode values:

Value	Name	Description	Project
1	URB_WRITE_OWORD **[Default]**	URB Oword Write message	All

VEBOX_STATE

VEBOX_STATE

Project:	CHV, BSW
Source:	VideoEnhancementCS
Length Bias:	2

This command controls the internal functions of the VEBOX. This command has a set of indirect state buffers:

- DN/DI state
- IECP general state
- IECP Gamut Expansion/Compression state
- IECP Gamut Vertex Table state
- Capture Pipe state

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:24	**Command OpCode** Default Value: 4h VEBOX Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 2h Format: OpCode
	15:12	**Reserved**
	11:0	**DWord Length** Format: =n Total Length - 2 Value: Ah / Name: / Description: (Excludes DWords 0, 1)
1	31:25	**State Surface Control Bits**
	24:23	**Reserved**

VEBOX_STATE

	22	**Reserved**
	21	**Reserved**
	20	**Reserved**
	19:15	**Reserved**
	14	**Single Slice VEBOX Enable** Project: CHV, BSW For products that have 2 entire VEBOXes that automatically split the frame, this enable emulates a 1 VEBOX product, running at 1/2 speed and only outputting a single set of per command statistics.
	13	**Hot Pixel Filtering Enable** Enables hot pixel detection/filtering.
	12	**Alpha Plane Enable** Enables the reading of an independent Alpha plane. Mutually exclusive with Vignette Enable. If **Alpha from State Select** is set it overrides this bit. **Programming Notes** IECP must also be enabled and output format must have alpha if this bit is enabled. Should be 0 if Alpha from State Select is 1.
	11	**Vignette Enable** Enables Vignette Correction surface read and correction in IECP. Mutually exclusive with Alpha Plane Enable. **Programming Notes** / **Project** Demosaic must also be enabled if this bit is enabled. / CHV, BSW
	10	**Demosaic Enable** The Demosaic will be used, and White balance statistics will be gathered. The Capture Pipe State Table will be read. This bit is mutually exclusive with **DI Enable.** **Programming Notes** IECP must also be enabled if this bit is enabled.
	9:8	**DI Output Frames** Indicates which frames to output in DI mode. **Value** / **Name** 00b / Output Both Frames 01b / Output Previous Frame Only 10b / Output Current Frame Only **Programming Notes** Field is ignored if DI Enable = 0. If Previous Frame Only or Current Frame Only are selected, then the **LACE Single Histogram Set** must not try to collect a histogram from the disabled frame. Field must be programmed to 10 (Output Current Frame Only) for DI First Frame.

VEBOX_STATE

	7	**444 -> 422 Downsample Method**
	6	**422 -> 420 Downsample Method**
	5	**DN/DI First Frame**
	4	**DI Enable**

Bit 7: 444 -> 422 Downsample Method

Value	Name
1	Average horizontally aligned chromas
0	Drop right chroma of the pair **[Default]**

Programming Notes

444->422	422->420	Description
0	0	No averaging, only down sampling
0	1	Not Supported
1	0	Only Horizontal averaging
1	1	Horizontal and Vertical averaging

Bit 6: 422 -> 420 Downsample Method

Project:	CHV, BSW

Value	Name
1	Average vertically aligned chromas
0	Drop lower chroma of the pair **[Default]**

Programming Notes

To enable averaging in case of 420 (NV12/P016) output formats, 444->422 and 422->420 should be set.

Bit 5: DN/DI First Frame

Format:	Enable

Indicates that this is the first frame of the stream, so previous clean is not available.

Value	Name
0	Not first field; previous clean surface state is valid
1	First field; previous clean surface state is invalid

Programming Notes

If both DN and DI are disabled, this bit must be 0.

Bit 4: DI Enable

Format:	Enable

Deinterlacer is bypassed if this is disabled: the output is the same as the input (same as a 2:2 cadence). FMD and STMM are not calculated and the values in the response message are 0.

Value	Name
0	Do not calculate DI
1	Calculate DI

VEBOX_STATE

	3	**DN Enable** Format: Enable Denoise is bypassed if this is low - BNE is still calculated and output, but the denoised fields are not. VDI does not read in the denoised previous frame but uses the pointer for the original previous frame. **Value** / **Name** 0 — Do not denoise frame 1 — Denoise frame **Programming Notes** If DN and/or Hotpixel are the only functions enabled then the only output is the Denoised Output which is the same surface format as the input. To get a format conversion with DN only, enable the Global IECP bit, but disable all the individual functions. The IECP output uses the output surface format.
	2	**Reserved**
	1	**Color Gamut Compression Enable** Indicates if the Gamut Compression feature is enabled. If set then the Gamut State will be read. VEB_VERTEXTABLE_STATE is only needed if this bit is set.
	0	**Color Gamut Expansion Enable** Indicates if the Gamut Expansion feature is enabled. If set then the Gamut State will be read.
2	31:12	**DN/DI State Pointer Low** Format: GraphicAddress[31:12] Bits 31:12 of the starting address of the DN/DI State buffer. This points to a buffer containing the 10 Dwords of the DN/DI state.
	11:0	**Reserved** Format: MBZ
3	31:16	**Reserved** Format: MBZ
	15:0	**DN/DI State Pointer High** Format: GraphicAddress[47:32] Bits 47:32 of the starting address of the DN/DI State Buffer.
4	31:12	**Reserved**
	11:0	**Reserved** Format: MBZ
5	31:16	**Reserved** Format: MBZ

VEBOX_STATE		
	15:0	**Reserved**
6	31:12	**Gamut State Pointer Low** Format: GraphicAddress[31:12] Bits 31:12 of the starting address of the Gamut State buffer. This points to a buffer containing the 42 Dwords of Gamut Compression / Gamut Expansion state.
	11:0	**Reserved** Format: MBZ
7	31:16	**Reserved** Format: MBZ
	15:0	**Gamut State Pointer High** Format: GraphicAddress[47:32] Bits 47:32 of the starting address of the Gamut State Buffer.
8	31:12	**Vertex Table State Pointer Low** Format: GraphicAddress[31:12] Bits 31:12 of the starting address of the Vertex Table. This points to a buffer containing the 512 Dwords of the Gamut Compression Vertex Table.
	11:0	**Reserved** Format: MBZ
9	31:16	**Reserved** Format: MBZ
	15:0	**Vertex Table State Pointer High** Format: GraphicAddress[47:32] Bits 47:32 of the starting address of the Vertex State Buffer.
10	31:12	**Capture Pipe State Pointer Low** Format: GraphicAddress[31:12] Bits 31:12 of the starting address of the Capture Pipe State Table. This points to a buffer containing the X Dwords of the Capture Pipe State.
	11:0	**Reserved** Format: MBZ
11	31:16	**Reserved** Format: MBZ
	15:0	**Capture Pipe State Pointer High** Format: GraphicAddress[47:32] Bits 47:32 of the starting address of the Capture Pipe State Table.

VEBOX_SURFACE_STATE

VEBOX_SURFACE_STATE	
Project:	CHV, BSW
Source:	VideoEnhancementCS
Length Bias:	2

Description
The input and output data containers accessed are called "surfaces". Surface state is sent to VEBOX via an inline state command rather than using binding tables. SURFACE_STATE contains the parameters defining each surface to be accessed, including its size, format, and offsets to its subsurfaces. The surface's base address is in the execution command. Despite having multiple input and output surfaces, we limit the number of surface states to one for input surfaces and one for output surfaces. The other surfaces are derived from the input/output surface states.
The Current Frame Input surface uses the Input SURFACE_STATE
The Previous Denoised Input surface uses the Input SURFACE_STATE. (For 12-bit Bayer pattern inputs this will be 8-bit.)
The Current Denoised Output surface uses the Input SURFACE_STATE. (For 12-bit Bayer pattern inputs this will be 8-bit.)
The STMM/Noise History Input surface uses the Input SURFACE_STATE with Tile-Y and Width/Height a multiple of 4.
The STMM/Noise History Output surface uses the Input SURFACE_STATE with Tile-Y and Width/Height a multiple of 4.
The Current Deinterlaced/IECP Frame Output surface uses the Output SURFACE_STATE.
The Previous Deinterlaced/IECP Frame Output surface uses the Output SURFACE_STATE.
The FMD per block output / per Frame Output surface uses the Linear SURFACE_STATE (see note below).
The Alpha surface uses the Linear A8 SURFACE_STATE with Width/Height equal to Input Surface. Pitch is width rounded to next 64.
The Vignette Correction surface uses the Linear 16-bit SURFACE_STATE with: Width = 4 * ((Input Width 3)/4) Height = ((Input Height 3)/4) Pitch in bytes is (vignette width*2) rounded to next 64.
The STMM height is the same as the Input Surface height except when the input **Surface Format** is Bayer Pattern and the **Bayer Pattern Offset** is 10 or 11, in which case the height is the input height + 4. For Bayer pattern inputs when the **Bayer Pattern Offset** is 10 or 11, the Current Denoised Output/Previous Denoised Input will also have a height which is the input height + 4. For Bayer pattern inputs only the Current Denoised Output/Previous Denoised Input are in Tile-Y.
The linear surface for FMD statistics is linear (not tiled). The height of the per block statistics is (Input Height +3)/4 - the Input Surface height in pixels is rounded up to the next even 4 and divided by 4. The width of the per block section in bytes is equal to the width of the Input Surface in pixels rounded up to the next 16 bytes. The pitch of the per block section in bytes is equal to the width of the Input Surface in pixels rounded up to the next 64 bytes.

VEBOX_SURFACE_STATE

The STMM surfaces must be identical to the Input surface except for the tiling mode must be Tile-Y and the pitch must be legal for Tile-Y (increased to the next larger legal pitch). If the input surface is packed (Surface Format from 0 to 3 for DN/DI) or 12/10-bit Bayer Pattern then the pitch for the STMM surface is 1/2 the pitch of the input surface (rounded up to the next larger legal Tile-Y pitch). The width and height must be a multiple of 4 rounded up from the input height.

Programming Notes
VEBOX may write to memory between the surface width and the surface pitch for output surfaces.
For 8bit Alpha input, when converting to 16bit output it is padded with 8bit zeros in the LSB.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Media Command Pipeline** Default Value: 2h Media Format: OpCode
	26:24	**Media Command OpCode** Default Value: 4h VEBOX Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h VEBOX Format: OpCode
	20:16	**SubOpcode B** Default Value: 0h VEBOX Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Total Length - 2 Value | Name | Description | Project 4h | DWORD_COUNT_n **[Default]** | (Excludes DWords 0, 1) | CHV, BSW
1	31:1	**Reserved**
	0	**Surface Identification** Specifies which set of surfaces this command refers to: Value | Name 1 | Output surface (all except the Denoised Current output surface) 0 | Input surface and Denoised Current Output Surface

VEBOX_SURFACE_STATE

2 | **31:18** | **Height**

Format:	U14

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

Value	Name	Description	Exists If
[15, 16383]		representing heights [16,16384]	
[15, 8191]			//Scalar Enabled - For Input surface only
[63, 2047]			//Scalar + SFC Enabled - For Input surface only

Programming Notes
Height (field value + 1) must be a multiple of 2 for PLANAR_420 surfaces. **Height** (field value +1) must be a multiple of 2 when the deinterlace function is enabled (field mode) or when the denoise function is enabled with **Progressive DN** = 0. It must be a multiple of 4 when interleaved deinterlace/denoise and PLANAR_420 are both being used. **VEBOX** supports a minimum height of 16.
Height (field value + 1) must be a multiple of 2 for Bayer surfaces.

17:4 | **Width**

Format:	U14

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

Value	Name	Description	Exists If
[63,16383]		representing widths [64,16384]	
[63,8191]			//Scalar Enabled - For Input surface only
[63,2047]			//Scalar and SFC Enabled - For Input Surface only

Programming Notes
The Width specified by this field multiplied by the pixel size in bytes must be less than or equal to the surface pitch (specified in bytes via the **Surface Pitch** field). **Width** (field value + 1) must be a multiple of 2 for PLANAR_420, PLANAR_422, and all YCRCB_* surfaces, and must be a multiple of 4 for PLANAR_411 surfaces. **VEBOX** supports a minimum width of 64

3:0 | **Reserved**

3 | **31:28** | **Surface Format**

Project:	CHV, BSW
Format:	U4

Specifies the format of the surface. All of the Y and G channels will use table 0 and all of the Cr/Cb/R/B channels will use table 1.

VEBOX_SURFACE_STATE

Value	Name	Description
4	PLANAR_420_8	NV12 with Interleave Chroma set
14	Bayer pattern	Demosaic input only
15	Reserved	

27 **Interleave Chroma**

Project:	CHV, BSW
Format:	Enable

This field indicates that the chroma fields are interleaved in a single plane rather than stored as two separate planes. This field is only used for PLANAR surface formats.

26:25 **Bayer Pattern Offset**

Project:	CHV, BSW

Specifies the starting pixel offset for the Bayer pattern used for Capture Pipe.

Value	Name
00b	Pixel at X=0, Y=0 is Blue
01b	Pixel at X=0, Y=0 is Red
10b	Pixel at X=0, Y=0 is Green, Pixel at X=1, Y=0 is Red
11b	Pixel at X=0, Y=0 is Green, Pixel at X=1, Y=0 is Blue

24 **Bayer Pattern Format**

Project:	CHV, BSW

Specifies the format of the Bayer Pattern:

Value	Name	Project
0b	8-bit input at a 8-bit stride	
1b	12 or 10-bit input at a 16-bit stride. Valid data is in the MSBs	CHV, BSW

23:21 **Reserved**

Project:	
Format:	MBZ

20 **Reserved**

Project:	CHV, BSW
Format:	MBZ

19:3 **Surface Pitch**

Format:	U17 pitch in (Bytes - 1)

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

Value	Name	Description
[63, 131071]	For other linear surfaces	[64B, 128KB]
[511, 131071]	For X-tiled surface	[512B, 128KB] = [1tile, 256 tiles]
[127, 131071]	For Y-tiled surfaces	[128B,128KB] = [1 tile, 1024 tiles]

VEBOX_SURFACE_STATE

DWord	Bit	Description
		Programming Notes For tiled surfaces, the pitch must be a multiple of the tile width. For linear surfaces, the pitch must be a multiple of 64. 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.
	2	**Half Pitch for Chroma** Format: Enable This field indicates that the chroma plane(s) will use a pitch equal to half the value specified in the **Surface Pitch** field. This field is only used for PLANAR surface formats. **Programming Notes** Must be programmed to Zero always as this field is not used
	1	**Tiled Surface** Format: Boolean This field specifies whether the surface is tiled. Value / Name / Description: 1 / True / Tiled 0 / False / Linear **Programming Notes** Linear surfaces can be mapped to Main Memory (uncached) or System Memory (cacheable, snooped). Tiled surfaces can only be mapped to Main Memory. The corresponding cache(s) must be invalidated before a previously accessed surface is accessed again with an altered state of this bit.
	0	**Tile Walk** Format: 3D_TileWalk This field specifies the type of memory tiling (XMajor or YMajor) employed to tile this surface. See *Memory Interface Functions* for details on memory tiling and restrictions. This field is ignored when the surface is linear. Value / Name: 0 / TILEWALK_XMAJOR 1 / TILEWALK_YMAJOR **Programming Notes** The corresponding cache(s) must be invalidated before a previously accessed surface is accessed again with an altered state of this bit.
4	31:29	**Reserved**
	28:16	**X Offset for U** Format: U13 Pixel Offset This field must be zero for the VEBOX surface formats

VEBOX_SURFACE_STATE			
	15	**Reserved**	
		Format:	MBZ
	14:0	**Y Offset for U**	
		Format:	U15 Row Offset
		This field specifies the veritical offset in rows from the **Surface Base Address** to the start (origin) of the U(Cb) plane or the interleaved UV plane if **Interleave Chroma** is enabled. This field is only used for PLANAR surface formats.	
		Programming Notes	
		This field must indicate an even number (bit 0 = 0). This field must be evenly divisible by 4 for Tile-Y surfaces (so the offset points to the start of a cache line) For Planar formats, if the surface is in YS or YF tile modes, the Y Offset for U should be an integral multiple of the Tile height of the Luma plane	
5	31:29	**Reserved**	
		Format:	MBZ
	28:16	**X Offset for V**	
		Format:	U13 Pixel Offset
		This field must be zero for the VEBOX surface formats.	
	15	**Reserved**	
		Format:	MBZ
	14:0	**Y Offset for V**	
		Format:	U15 Row Offset
		This field specifies the veritical offset in rows from the **Surface Base Address** to the start (origin) of the V(Cr) plane. This field is only used for PLANAR surface formats with **Interleave Chroma** disabled.	
		Programming Notes	
		This field must indicate an even number (bit 0 = 0). This field must be evenly divisible by 4 for Tile-Y surfaces (so the offset points to the start of a cache line). For Planar formats, if the surface is in YS or YF tile modes, the Y Offset for V should be an integral multiple of the Tile height of the Luma plane	
6..7	31:0	**Reserved**	
		Project:	CHV, BSW
		Format:	MBZ

Wait Notification

wait - Wait Notification

Project:	CHV, BSW
Source:	Eulsa
Length Bias:	4

The wait instruction evaluates the value of the notification count register nreg. If nreg is zero, thread execution is suspended and the thread is put in 'wait_for_notification' state. If nreg is not zero (i.e., one or more notifications have been received), nreg is decremented by one and the thread continues executing on the next instruction. If a thread is in the 'wait_for_notification' state, when a notification arrives, the notification count register is incremented by one. As the notification count register becomes nonzero, the thread wakes up to continue execution and at the same time the notification register is decremented by one. If only one notification arrived, the notification register value becomes zero. However, during the above mentioned time period, it is possible that more notifications may arrive, making the notification register nonzero again. When multiple notifications are received, software must use wait instructions to decrement notification count registers for each notification. Notification register n0.0:ud is for thread to thread communication (via the Message Gateway shared function) and n0.1:ud for host to thread communication (through MMIO registers). See the Message Gateway chapter for thread-thread communication and the Debug chapter for host-to-thread communication.

Format: wait (exec_size) nreg

Restriction
src0 and dst must be n0.0, n0.1, or n0.2.
Execution size must be 1 as the notification registers are scalar.
Predication is not allowed.
Two back-to-back wait instructions are not allowed. At minimum, a nop instruction must be inserted between two wait instructions

Syntax
wait (1) n#

Pseudocode
N/A

Predication	Conditional Modifier	Saturation	Source Modifier
N	N	N	N

Src Types	Dst Types
UD	UD

DWord	Bit	Description
0	127:64	**Sources** Exists If: ([Operand Control][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG [CHV, BSW]
	127:64	**Sources** Exists If: ([Operand Control][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32 [CHV, BSW]
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header** Format: EU_INSTRUCTION_HEADER [CHV, BSW]

While

while - While

Project:	CHV, BSW
Source:	EuIsa
Length Bias:	4

Description
The while instruction marks the end of a do-while block. The instruction first evaluates the loop termination condition for each channel based on the current channel enables and the predication flags specified in the instruction. If any channel has not terminated, a branch is taken to a destination address specified in the instruction, and the loop continues for those channels. Otherwise, execution continues to the next instruction.Id point to the first instruction with the do label of the do-while block of code. It should be a negative number for the backward referencing. In GEN binary, JIP is at location dst and must be of type W (signed word integer). If SPF is ON, none of the PcIP are updated.
The following table describes the 32-bit jump target offset JIP. JIP is a signed 32-bit number, added to IP pre-increment, and should point to the first instruction with the do label of the do-while block of code. It should be a negative number for the backward referencing. In GEN binary, JIP is at location src1 and must be of type D (signed dword integer).
Format: [(pred)] while (exec_size) JIP

Restriction
The execution size must be the same for the while instruction and any break and cont instructions of the same code block.

Syntax	Project
[(pred)] while (exec_size) imm32	CHV, BSW

Pseudocode
Evaluate(WrEn); for (n = 0; n < 32; n++) { if (WrEn.chan[n]) { PcIP[n] = IP + JIP; } else { PcIP[n] = IP + 1; } } if (| PMask == 1) { // any enabled channel true Jump(IP + JIP); }

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

DWord	Bit	Description
0..3	127:96	**JIP** Project: CHV, BSW Format: S31 Jump Target Offset. The relative offset in bytes if a jump is taken for the instruction.
	95	**Source 0 Address Immediate [9] Sign Bit** Project: CHV, BSW

while - While

	94:91	**Src1.SrcType**	
		Project:	CHV, BSW
		Format:	SrcType [CHV, BSW]
	90:89	**Src1.RegFile**	
		Project:	CHV, BSW
		Format:	RegFile [CHV, BSW]
	88:64	**Source 0**	
		Exists If:	(Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align16')
		Format:	EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN16 [CHV, BSW]
	88:64	**Source 0**	
		Exists If:	(Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1')
		Format:	EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN1 [CHV, BSW]
	63:32	**Operand Control**	
		Format:	EU_INSTRUCTION_OPERAND_CONTROLS [CHV, BSW]
	31:0	**Header**	
		Format:	EU_INSTRUCTION_HEADER [CHV, BSW]

XY_COLOR_BLT

XY_COLOR_BLT

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

COLOR_BLT is the simplest BLT operation. It performs a color fill to the destination (with a possible ROP). The only operand is the destination operand which is written dependent on the raster operation. The solid pattern color is stored in the pattern background register.
This instruction is optimized to run at the maximum memory write bandwidth.
The typical (and fastest) Raster operation code = F0 which performs a copy of the pattern background register to the destination.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 50h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. Value / Name: 1xb: Write Alpha Channel x1b: Write RGB Channel
	19:12	**Reserved** Format: MBZ
	11	**Tiling Enable** Value / Name / Description: 0b: Tiling Disabled (Linear Blit) 1b: Tiling Enabled: Tile-X or Tile-Y
	10:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 05h
1 BR13	31	**Reserved** Format: MBZ
	30	**Clipping Enabled** Value / Name

XY_COLOR_BLT

DWord	Bit	Description
		0b: Disabled 1b: Enabled
	29:26	**Reserved** Format: MBZ
	25:24	**Color Depth** **Value** / **Name** 00b: 8 Bit Color 01b: 16 Bit Color(565) 10b: 16 Bit Color(1555) 11b: 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4 BR09	31:0	**Destination Base Address** Format: GraphicsAddress[31:0] Base address of the destination surface: X=0, Y=0. Lower 32bits of the 48bit addressing. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes.When Tiling is not enabled, this address should be CL (64byte) aligned.
5 BR27	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.
	15:0	**Destination Base Address High** Format: GraphicsAddress[47:32] Should be programmed with the upper 16bits of the 48bit addressing.
6 BR16	31:0	**Solid Pattern Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]

XY_FULL_BLT

XY_FULL_BLT

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

The full BLT is the most comprehensive BLT instruction. It provides the ability to specify all 3 operands: destination, source, and pattern. The source and pattern operands are the same bit width as the destination operand.

The source and destination operands may overlap, which means that the X and Y directions can be either forward or backwards. The BLT Engine takes care of all situations. The base addresses plus the X and Y coordinates determine if there is an overlap between the source and destination operands. If the base addresses of the source and destination are the same and the Source X1 is less than Destination X1, then the BLT Engine performs the accesses in the X-backwards access pattern. There is no need to look for an actual overlap. If the base addresses are the same and Source Y1 is less than Destination Y1, then the scan line accesses start at Destination Y2 with the corresponding source scan line and the strides are subtracted for every scan line access.

All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.

The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 55h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. Value / Name: 00b — **[Default]** 1xb — Write Alpha Channel x1b — Write RGB Channel
	19:16	**Reserved**
	15	**Src Tiling Enable** Value / Name / Description: 0b — Tiling Disabled (Linear Blit) — 1b — Tiling Enabled — [CHV, BSW]: Tile-X or Tile-Y.
	14:12	**Pattern Horizontal Seed** Pixel of the scan line to start on corresponding to DST X=0.

XY_FULL_BLT

DWord	Bit	Description
	11	**Dest Tiling Enable**
		Value: 0b — Name: Tiling Disabled (Linear Blit) — Description:
		Value: 1b — Name: Tiling Enabled — Description: [CHV, BSW] [CHV, BSW]: Tile-X or Tile-Y.
	10:8	**Pattern Vertical Seed** Starting scan line of the 8x8 pattern corresponding to DST Y=0.
	7:0	**DWord Length** Default Value: 0Ah
1 BR13	31	**Reserved**
	30	**Clipping Enabled** 0b: Disabled 1b: Enabled
	29:26	**Reserved**
	25:24	**Color Depth** 00b: 8 Bit Color 01b: 16 Bit Color(565) 10b: 16 Bit Color(1555) 11b: 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4 BR09	31:0	**Destination Base Address** Format: GraphicsAddress[31:0] Base address of the destination surface: X=0, Y=0. Lower 32bits of the 48bit addressing. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.

XY_FULL_BLT		
5 BR27	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.
	15:0	**Destination Base Address High** Format: GraphicsAddress[47:32] Should be programmed with the upper 16bits of the 48bit addressing.
6 BR11	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.
	15:0	**Source Pitch (double word aligned and signed) and in DWords** 2's complement. For Tiled Src (bit 15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
7 BR26	31:16	**Source Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Source X1 Coordinate (Left)** 16 bit signed number.
8 BR12	31:0	**Source Address** Format: GraphicsAddress[31:0] Base address of the destination surface: X=0, Y=0. Lower 32bits of the 48bit addressing. When Tiling is enabled (Bit_15 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL(64byte) aligned.
9 BR28	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.
	15:0	**Source Address High** Format: GraphicsAddress[47:32] Should be programmed with the upper 16bits of the 48bit addressing.
10 BR15	31:0	**Pattern Base Address** Format: GraphicsAddress[31:0] (28:06 are implemented) (Note no NPO2 change here). Lower 32bits of the 48bit addressing. The pattern data must be located in linear memory. The Pattern Base Address programmed, must always be Cache Line (64byte) aligned.
11 BR29	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.
	15:0	**Pattern Base Address High** Format: GraphicsAddress[47:32] Should be programmed with the upper 16bits of the 48bit addressing.

XY_FULL_IMMEDIATE_PATTERN_BLT

XY_FULL_IMMEDIATE_PATTERN_BLT

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

The full BLT is the most comprehensive BLT instruction. It provides the ability to specify all 3 operands: destination, source, and pattern. The source and immediate pattern operands are the same bit width as the destination operand. The immediate data sizes are 64 bytes (16 DWs), 128 bytes (32 DWs), or 256 (64 DWs) for 8, 16, and 32 bpp color patterns. DWL indicates the total number of Dwords of immediate data.
The source and destination operands may overlap, which means that the X and Y directions can be either forward or backwards. The BLT Engine takes care of all situations. The base addresses plus the X and Y coordinates determine if there is an overlap between the source and destination operands. If the base addresses of the source and destination are the same and the Source X1 is less than Destination X1, then the BLT Engine performs the accesses in the X-backwards access pattern. There is no need to look for an actual overlap. If the base addresses are the same and Source Y1 is less than Destination Y1, then the scan line accesses start at Destination Y2 with the corresponding source scan line and the strides are subtracted for every scan line access.
All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.
The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 74h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp.
	19:16	**Reserved** Format: MBZ
	15	**Src Tiling Enable**

32bpp Byte Mask values:

Value	Name
00b	**[Default]**
1xb	Write Alpha Channel
x1b	Write RGB Channel

Src Tiling Enable values:

Value	Name	Description
0b	Tiling Disabled (Linear)	

XY_FULL_IMMEDIATE_PATTERN_BLT

DWord	Bit	Description
		1b — Tiling Enabled — [CHV, BSW] [CHV, BSW]: Tile-X or Tile-Y.
	14:12	**Pattern Horizontal Seed** (pixel of the scan line to start on corresponding to DST X=0)
	11	**Dest Tiling Enable** Value / Name / Description: 0b — Tiling Disabled (Linear Blit) 1b — Tiling Enabled — [CHV, BSW] [CHV, BSW]: Tile-X or Tile-Y.
	10:8	**Pattern Vertical Seed** Starting scan line of the 8x8 pattern corresponding to DST Y=0.
	7:0	**DWord Length** Default Value: 08h Excludes DWORD 0,1 08 + DWL = (Number of Immediate double words)h
1 BR13	31	**Reserved** Format: MBZ
	30	**Clipping Enabled** Value / Name: 0b — Disabled 1b — Enabled
	29:26	**Reserved** Format: MBZ
	25:24	**Color Depth** Value / Name: 00b — 8 Bit Color 01b — 16 Bit Color(565) 10b — 16 Bit Color(1555) 11b — 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)**

XY_FULL_IMMEDIATE_PATTERN_BLT		
		16 bit signed number.
4 BR9	31:0	**Destination Base Address** Format: GraphicsAddress[31:0] Base address of the destination surface: X=0, Y=0. Lower 32bits of the 48bit addressing. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.
5 BR27	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.
	15:0	**Destination Base Address High** Format: GraphicsAddress[47:32] Should be programmed with the upper 16bits of the 48bit addressing.
6 BR11	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.
	15:0	**Source Pitch (double word aligned and signed) and in DWords** 2's complement. For Tiled Src (bit 15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
7 BR26	31:16	**Source Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Source X1 Coordinate (Left)** 16 bit signed number.
8 BR12	31:0	**Source Address** Format: GraphicsAddress[31:0] Base address of the destination surface: X=0, Y=0. Lower 32bits of the 48bit addressing. When Src Tiling is enabled (Bit_15 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.
9 BR28	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.
	15:0	**Source Address High** Format: GraphicsAddress[47:32] Should be programmed with the upper 16bits of the 48bit addressing.
10..n	31:0	**Immediate Data 0**

XY_FULL_MONO_PATTERN_BLT

XY_FULL_MONO_PATTERN_BLT

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

The full BLT is the most comprehensive BLT instruction. It provides the ability to specify all 3 operands: destination, source, and pattern. The pattern operand is monochrome and the source operand is the same bit width as the destination operand.

The source and destination operands may overlap, which means that the X and Y directions can be either forward or backwards. The BLT Engine takes care of all situations. The base addresses plus the X and Y coordinates determine if there is an overlap between the source and destination operands. If the base addresses of the source and destination are the same and the Source X1 is less than Destination X1, then the BLT Engine performs the accesses in the X-backwards access pattern. There is no need to look for an actual overlap. If the base addresses are the same and Source Y1 is less than Destination Y1, then the scan line accesses start at Destination Y2 with the corresponding source scan line and the strides are subtracted for every scan line access.

The monochrome pattern transparency mode indicates whether to use the pattern background color or de-assert the write enables when the bit in the source is 0. When the source bit is 1, then the pattern foreground color is used in the ROP operation.

All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.

The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.

Setting both Solid Pattern Select =1 and Mono Pattern Transparency = 1 is mutually exclusive. The device implementation results in NO PIXELs DRAWN.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 57h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. Value / Name: 00b — **[Default]** 1xb — Write Alpha Channel x1b — Write RGB Channel
	19:16	**Reserved**

<table>
<tr><th colspan="3">XY_FULL_MONO_PATTERN_BLT</th></tr>
<tr><td rowspan="5"></td><td>15</td><td>Src Tiling Enable
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0b</td><td>Tiling Disabled (Linear Blit)</td><td></td></tr>
<tr><td>1b</td><td>Tiling Enabled</td><td>[CHV, BSW] [CHV, BSW]: Tile-X or Tile-Y.</td></tr>
</table></td></tr>
<tr><td>14:12</td><td>Pattern Horizontal Seed
(pixel of the scan line to start on corresponding to DST X=0)</td></tr>
<tr><td>11</td><td>Dest Tiling Enable
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0b</td><td>Tiling Disabled (Linear Blit)</td><td></td></tr>
<tr><td>1b</td><td>Tiling Enabled</td><td>[CHV, BSW] [CHV, BSW]: Tile-X or Tile-Y.</td></tr>
</table></td></tr>
<tr><td>10:8</td><td>Pattern Vectical Seed
Starting scan line of the 8x8 pattern corresponding to DST Y=0.</td></tr>
<tr><td>7:0</td><td>DWord Length
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0Ch</td><td></td></tr>
</table></td></tr>
<tr><td rowspan="7">1

BR13</td><td>31</td><td>Solid Pattern Select
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0</td><td>No Solid Pattern</td></tr>
<tr><td>1</td><td>Solid Pattern</td></tr>
</table></td></tr>
<tr><td>30</td><td>Clipping Enabled
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0b</td><td>Disabled</td></tr>
<tr><td>1b</td><td>Enabled</td></tr>
</table></td></tr>
<tr><td>29</td><td>Reserved
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td>28:27</td><td>Mono Source Transparency Mode
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0</td><td>Use Background</td></tr>
<tr><td>1</td><td>Transparency Enabled</td></tr>
</table></td></tr>
<tr><td>26</td><td>Reserved</td></tr>
<tr><td>25:24</td><td>Color Depth
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>00b</td><td>8 Bit Color</td></tr>
<tr><td>01b</td><td>16 Bit Color(565)</td></tr>
<tr><td>10b</td><td>16 Bit Color(1555)</td></tr>
<tr><td>11b</td><td>32 Bit Color</td></tr>
</table></td></tr>
</table>

XY_FULL_MONO_PATTERN_BLT		
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4 BR09	31:0	**Destination Base Address** Format: GraphicsAddress[31:0] Base address of the destination surface: X=0, Y=0. Lower 32bits of the 48bit addressing. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.
5 BR27	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.
	15:0	**Destination Base Address High** Format: GraphicsAddress[47:32] Should be programmed with the upper 16bits of the 48bit addressing.
6 BR11	31:16	**Reserved** Format: MBZ
	15:0	**Source Pitch (double word aligned and signed) and in DWords** 2's complement. For Tiled Src (bit 15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
7 BR26	31:16	**Source Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Source X1 Coordinate (Left)** 16 bit signed number.
8 BR12	31:0	**Source Address** Format: GraphicsAddress[31:0] (base address of the source surface: X=0, Y=0). Lower 32bits of the 48bit addressing. When Src Tiling is enabled (Bit 15 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.

XY_FULL_MONO_PATTERN_BLT		
9 BR28	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.
	15:0	**Source Address High** Format: GraphicsAddress[47:32] Should be programmed with the upper 16bits of the 48bit addressing.
10 BR16	31:0	**Pattern Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
11 BR17	31:0	**Pattern Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
12 BR20	31:0	**Pattern Data 0** (least significant DW)
13 BR21	31:0	**Pattern Data 1** (most significant DW)

XY_FULL_MONO_PATTERN_MONO_SRC_BLT

XY_FULL_MONO_PATTERN_MONO_SRC_BLT

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

The full BLT provides the ability to specify all 3 operands: destination, source, and pattern. The pattern and source operands are monochrome.

The monochrome source transparency mode indicates whether to use the source background color or de-assert the write enables when the bit in the source is 0. When the source bit is 1, then the source foreground color is used in the ROP operation.

All non-text monochrome sources are word aligned. At the end of a scan line the monochrome source, the remaining bits until the next word boundary are ignored. The Monochrome source data bit position field [2:0] indicates which bit position within the first byte should be used as the first source pixel which corresponds to the destination X1 coordinate.

The monochrome pattern transparency mode indicates whether to use the pattern background color or de-assert the write enables when the bit in the pattern is 0. When the source bit is 1, then the pattern foreground color is used in the ROP operation. The monochrome source transparency mode works identical to the pattern transparency mode.

All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.

The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.

Setting both Solid Pattern Select =1 and Mono Pattern Transparency = 1 is mutually exclusive. The device implementation results in NO PIXELs DRAWN.

Negative Stride (= Pitch) is NOT ALLOWED.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="4">0

BR00</td><td>31:29</td><td>Client
<table><tr><td>Default Value:</td><td>02h 2D Processor</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td>28:22</td><td>Instruction Target(Opcode)
<table><tr><td>Default Value:</td><td>58h</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td>21:20</td><td>32bpp Byte Mask
This field is only used for 32bpp.
<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>[Default]</td></tr><tr><td>1xb</td><td>Write Alpha Channel</td></tr><tr><td>x1b</td><td>Write RGB Channel</td></tr></table></td></tr>
<tr><td>19:17</td><td>Monochrome source data bit position of the first pixel within a byte per scan line.</td></tr>
</table>

<table>
<tr><th colspan="3">XY_FULL_MONO_PATTERN_MONO_SRC_BLT</th></tr>
<tr><td></td><td>16:15</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>14:12</td><td>Pattern Horizontal Seed
(pixel of the scan line to start on corresponding to DST X=0)</td></tr>
<tr><td></td><td>11</td><td>Tiling Enable<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0b</td><td>Tiling Disabled (Linear Blit)</td><td></td></tr><tr><td>1b</td><td>Tiling Enabled</td><td>[CHV, BSW] [CHV, BSW]: Tile-X or Tile-Y.</td></tr></table></td></tr>
<tr><td></td><td>10:8</td><td>Pattern Vertical Seed
Starting scan line of the 8x8 pattern corresponding to DST Y = 0.</td></tr>
<tr><td></td><td>7:0</td><td>DWord Length<table><tr><th>Value</th><th>Name</th></tr><tr><td>0Ch</td><td></td></tr></table></td></tr>
<tr><td>1
BR13</td><td>31</td><td>Solid Pattern Select<table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>No Solid Pattern</td></tr><tr><td>1</td><td>Solid Pattern</td></tr></table></td></tr>
<tr><td></td><td>30</td><td>Clipping Enabled<table><tr><th>Value</th><th>Name</th></tr><tr><td>0b</td><td>Disabled</td></tr><tr><td>1b</td><td>Enabled</td></tr></table></td></tr>
<tr><td></td><td>29</td><td>Mono Source Transparency Mode<table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>Use Background</td></tr><tr><td>1</td><td>Transparency Enabled</td></tr></table></td></tr>
<tr><td></td><td>28</td><td>Mono Pattern Transparency Mode<table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>Use Background</td></tr><tr><td>1</td><td>Transparency Enabled</td></tr></table></td></tr>
<tr><td></td><td>27:26</td><td>Reserved</td></tr>
<tr><td></td><td>25:24</td><td>Color Depth<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>8 Bit Color</td></tr><tr><td>01b</td><td>16 Bit Color(565)</td></tr><tr><td>10b</td><td>16 Bit Color(1555)</td></tr><tr><td>11b</td><td>32 Bit Color</td></tr></table></td></tr>
</table>

XY_FULL_MONO_PATTERN_MONO_SRC_BLT		
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4 BR09	31:0	**Destination Base Address** Format: GraphicsAddress[31:0] Base address of the destination surface: X=0, Y=0. Lower 32bits of the 48bit addressing. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes.When Tiling is not enabled, this address should be CL (64byte) aligned.
5 BR27	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.
	15:0	**Destination Base Address High** Format: GraphicsAddress[47:32] Should be programmed with the upper 16bits of the 48bit addressing.
6 BR12	31:0	**Mono Source Address** Format: GraphicsAddress[31:0] (address corresponds to DST X1, Y1) (Note no NPO2 change here). Lower 32bits of the 48bit addressing. This Monosource Base Address programmed, must always be Cache Line (64byte) aligned.
7 BR28	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.
	15:0	**Mono Source Address High** Format: GraphicsAddress[47:32] Should be programmed with the upper 16bits of the 48bit addressing.
8 BR18	31:0	**Source Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]

XY_FULL_MONO_PATTERN_MONO_SRC_BLT		
9 BR19	31:0	**Source Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
10 BR16	31:0	**Pattern Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
11 BR17	31:0	**Pattern Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
12 BR20	31:0	**Pattern Data 0** (least significant DW)
13 BR21	31:0	**Pattern Data 1** (most significant DW)

XY_FULL_MONO_SRC_BLT

XY_FULL_MONO_SRC_BLT

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

The full BLT is the most comprehensive BLT instruction. It provides the ability to specify all 3 operands: destination, source, and pattern. The source operand is monochrome and the pattern operand is the same bit width as the destination.

The monochrome source transparency mode indicates whether to use the source background color or de-assert the write enables when the bit in the source is 0. When the source bit is 1, then the source foreground color is used in the ROP operation.

All non-text and non-immediate monochrome sources are word aligned. At the end of a scan line the monochrome source, the remaining bits until the next word boundary are ignored. The Monochrome source data bit position field [2:0] indicates which bit position within the first byte should be used as the first source pixel which corresponds to the Destination X1 coordinate.

All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.

The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.

Negative Stride (= Pitch) is NOT ALLOWED

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="6">0

BR00</td><td>31:29</td><td>Client<table><tr><td>Default Value:</td><td>02h 2D Processor</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td>28:22</td><td>Instruction Target(Opcode)<table><tr><td>Default Value:</td><td>56h</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td>21:20</td><td>32bpp Byte Mask
This field is only used for 32bpp.<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>[Default]</td></tr><tr><td>1xb</td><td>Write Alpha Channel</td></tr><tr><td>x1b</td><td>Write RGB Channel</td></tr></table></td></tr>
<tr><td>19:17</td><td>Monochrome source data bit position of the first pixel within a byte per scan line.</td></tr>
<tr><td>16:15</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>14:12</td><td>Pattern Horizontal Seed
(pixel of the scan line to start on corresponding to DST X=0)</td></tr>
</table>

<table>
<tr><th colspan="3">XY_FULL_MONO_SRC_BLT</th></tr>
<tr><td></td><td>11</td><td>Tiling Enable
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0b</td><td>Tiling Disabled (Linear Blit)</td><td></td></tr>
<tr><td>1b</td><td>Tiling Enabled</td><td>[CHV, BSW] [CHV, BSW]: Tile-X or Tile-Y.</td></tr>
</table></td></tr>
<tr><td></td><td>10:8</td><td>Pattern Vertical Seed
Starting scan line of the 8x8 pattern corresponding to DST Y = 0.</td></tr>
<tr><td></td><td>7:0</td><td>DWord Length
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0Ah</td><td></td></tr>
</table></td></tr>
<tr><td>1
BR13</td><td>31</td><td>Reserved
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>30</td><td>Clipping Enabled
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0b</td><td>Disabled</td></tr>
<tr><td>1b</td><td>Enabled</td></tr>
</table></td></tr>
<tr><td></td><td>29</td><td>Mono Source Transparency Mode
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0</td><td>Use Background</td></tr>
<tr><td>1</td><td>Transparency Enabled</td></tr>
</table></td></tr>
<tr><td></td><td>28:26</td><td>Reserved
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>25:24</td><td>Color Depth
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>00b</td><td>8 Bit Color</td></tr>
<tr><td>01b</td><td>16 Bit Color(565)</td></tr>
<tr><td>10b</td><td>16 Bit Color(1555)</td></tr>
<tr><td>11b</td><td>32 Bit Color</td></tr>
</table></td></tr>
<tr><td></td><td>23:16</td><td>Raster Operation</td></tr>
<tr><td></td><td>15:0</td><td>Destination Pitch in DWords
2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).</td></tr>
<tr><td>2
BR22</td><td>31:16</td><td>Destination Y1 Coordinate (Top)
16 bit signed number.</td></tr>
<tr><td></td><td>15:0</td><td>Destination X1 Coordinate (Left)
16 bit signed number.</td></tr>
<tr><td>3</td><td>31:16</td><td>Destination Y2 Coordinate (Bottom)
16 bit signed number.</td></tr>
</table>

XY_FULL_MONO_SRC_BLT		
BR23	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4 BR09	31:0	**Destination Base Address** Format: GraphicsAddress[31:0] Base address of the destination surface: X=0, Y=0. Lower 32bits of the 48bit addressing. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.
5 BR27	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.
	15:0	**Destination Base Address High** Format: GraphicsAddress[47:32] Should be programmed with the upper 16bits of the 48bit addressing.
6 BR12	31:0	**Mono Source Address** Format: GraphicsAddress[31:0] (address corresponds to DST X1, Y1) (Note no NPO2 change here). Lower 32bits of the 48bit addressing. This Monosource Base Address programmed, must always be Cache Line (64byte) aligned.
7 BR28	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.
	15:0	**Mono Source Address High** Format: GraphicsAddress[47:32] Should be programmed with the upper 16bits of the 48bit addressing.
8 BR18	31:0	**Source Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
9 BR19	31:0	**Source Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
10 BR15	31:0	**Pattern Base Address** Format: GraphicsAddress[31:0] (28:06 are implemented) (Note no NPO2 change here). Lower 32bits of the 48bit addressing. The pattern data must be located in linear memory. The Pattern Base Address programmed, must always be Cache Line (64byte) aligned.

XY_FULL_MONO_SRC_BLT		
11 BR29	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.
	15:0	**Pattern Base Address High** Format: GraphicsAddress[47:32] Should be programmed with the upper 16bits of the 48bit addressing.

XY_FULL_MONO_SRC_IMMEDIATE_PATTERN_BLT

XY_FULL_MONO_SRC_IMMEDIATE_PATTERN_BLT

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

The full BLT is the most comprehensive BLT instruction. It provides the ability to specify all 3 operands: destination, source, and pattern. The source operand is a monochrome and the immediate pattern operand is the same bit width as the destination. The immediate data sizes are 64 bytes (16 DWs), 128 bytes (32 DWs), or 256 (64DWs) for 8, 16, and 32 bpp color patterns. The monochrome source transparency mode indicates whether to use the source background color or de-assert the write enables when the bit in the source is 0. When the source bit is 1, then the source foreground color is used in the ROP operation. All non-text monochrome sources are word aligned. At the end of a scan line the monochrome source, the remaining bits until the next word boundary are ignored. The Monochrome source data bit position field [2:0] indicates which bit position within the first byte should be used as the first source pixel which corresponds to the destination X1 coordinate. All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation. The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8. Negative Stride (= Pitch) is NOT ALLOWED.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 75h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. Value / Name: 00b — **[Default]** 1xb — Write Alpha Channel x1b — Write RGB Channel
	19:17	**Monochrome source data bit position of the first pixel within a byte per scan line.**
	16:15	**Reserved** Format: MBZ
	14:12	**Pattern Horizontal Seed** (pixel of the scan line to start on corresponding to DST X=0)

<table>
<tr><th colspan="3">XY_FULL_MONO_SRC_IMMEDIATE_PATTERN_BLT</th></tr>
<tr><td rowspan="3"></td><td>11</td><td>Tiling Enable
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0b</td><td>Tiling Disabled (Linear Blit)</td><td></td></tr>
<tr><td>1b</td><td>Tiling Enabled</td><td>[CHV, BSW] [CHV, BSW]: Tile-X or Tile-Y.</td></tr>
</table></td></tr>
<tr><td>10:8</td><td>Pattern Vertical Seed
Starting scan line of the 8x8 pattern corresponding to DST Y=0.</td></tr>
<tr><td>7:0</td><td>DWord Length
<table>
<tr><td>Default Value:</td><td>08h Excludes DWORD 0,1</td></tr>
</table>
08 + DWL = (Number of Immediate double words)h</td></tr>
<tr><td rowspan="8">1

BR13</td><td>31</td><td>Reserved
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td>30</td><td>Clipping Enabled
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0b</td><td>Disabled</td></tr>
<tr><td>1b</td><td>Enabled</td></tr>
</table></td></tr>
<tr><td>29</td><td>Mono Source Transparency Mode
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0</td><td>Use Background</td></tr>
<tr><td>1</td><td>Transparency Enabled</td></tr>
</table></td></tr>
<tr><td>28:26</td><td>Reserved
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td>25:24</td><td>Color Depth
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>00b</td><td>8 Bit Color</td></tr>
<tr><td>01b</td><td>16 Bit Color(565)</td></tr>
<tr><td>10b</td><td>16 Bit Color(1555)</td></tr>
<tr><td>11b</td><td>32 Bit Color</td></tr>
</table></td></tr>
<tr><td>23:16</td><td>Raster Operation</td></tr>
<tr><td>15:0</td><td>Destination Pitch in DWords
2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).</td></tr>
<tr style="display:none"><td></td><td></td></tr>
<tr><td rowspan="2">2

BR22</td><td>31:16</td><td>Destination Y1 Coordinate (Top)
16 bit signed number.</td></tr>
<tr><td>15:0</td><td>Destination X1 Coordinate (Left)
16 bit signed number.</td></tr>
<tr><td>3</td><td>31:16</td><td>Destination Y2 Coordinate (Bottom)
16 bit signed number.</td></tr>
</table>

<table>
<tr><th colspan="3">XY_FULL_MONO_SRC_IMMEDIATE_PATTERN_BLT</th></tr>
<tr><td>BR23</td><td>15:0</td><td>Destination X2 Coordinate (Right)
16 bit signed number.</td></tr>
<tr><td>4
BR09</td><td>31:0</td><td>Destination Base Address<table><tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr></table>Base address of the destination surface: X=0, Y=0. Lower 32bits of the 48bit addressing. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.</td></tr>
<tr><td rowspan="2">5
BR27</td><td>31:16</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table>Should be programmed all 0's for 48bit addressing.</td></tr>
<tr><td>15:0</td><td>Destination Base Address High<table><tr><td>Format:</td><td>GraphicsAddress[47:32]</td></tr></table>Should be programmed with the upper 16bits of the 48bit addressing.</td></tr>
<tr><td>6
BR12</td><td>31:0</td><td>Mono Source Address<table><tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr></table>(address corresponds to DST X1, Y1) (Note no NPO2 change here). Lower 32bits of the 48bit addressing.
This Monosource Base Address programmed, must always be Cache Line (64byte) aligned.</td></tr>
<tr><td rowspan="2">7
BR28</td><td>31:16</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table>Should be programmed all 0's for 48bit addressing.</td></tr>
<tr><td>15:0</td><td>Mono Source Address High<table><tr><td>Format:</td><td>GraphicsAddress[47:32]</td></tr></table>Should be programmed with the upper 16bits of the 48bit addressing.</td></tr>
<tr><td>8
BR18</td><td>31:0</td><td>Source Background Color
8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]</td></tr>
<tr><td>9
BR19</td><td>31:0</td><td>Source Foreground Color
8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]</td></tr>
<tr><td>10..n</td><td>31:0</td><td>Immediate Data</td></tr>
</table>

XY_MONO_PAT_BLT

XY_MONO_PAT_BLT

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

MONO_PAT_BLT is used when we have no source and the monochrome pattern is not trivial (is not a solid color only). The monochrome pattern is loaded from the instruction stream.

All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.

The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.

The monochrome pattern transparency mode indicates whether to use the pattern background color or de-assert the write enables when the bit in the pattern is 0. When the pattern bit is 1, then the pattern foreground color is used in the ROP operation.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 52h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. Value / Name: 00b — **[Default]** 1xb — Write Alpha Channel x1b — Write RGB Channel
	19:15	**Reserved** Format: MBZ
	14:12	**Pattern Horizontal Seed** Pixel of the scan line to start on corresponding to DST X=0.
	11	**Tiling Enable** Value / Name / Description: 0b — Tiling Disabled (Linear Blit) — 1b — Tiling Enabled — [CHV, BSW]: Tile-X or Tile-Y.
	10:8	**Pattern Vertical Seed** Scan line of the 8x8 pattern to start on corresponding to DST Y=0.

<table>
<tr><th colspan="3">XY_MONO_PAT_BLT</th></tr>
<tr><td></td><td>7:0</td><td>DWord Length
<table><tr><th>Value</th><th>Name</th></tr><tr><td>08h</td><td></td></tr></table></td></tr>
<tr><td rowspan="8">1
BR13</td><td>31</td><td>Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>30</td><td>Clipping Enabled
<table><tr><th>Value</th><th>Name</th></tr><tr><td>0b</td><td>Disabled</td></tr><tr><td>1b</td><td>Enabled</td></tr></table></td></tr>
<tr><td>29</td><td>Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>28</td><td>Mono Pattern Transparency Mode
<table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>Use Background</td></tr><tr><td>1</td><td>Transparency Enabled</td></tr></table></td></tr>
<tr><td>27:26</td><td>Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>25:24</td><td>Color Depth
<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>8 Bit Color</td></tr><tr><td>01b</td><td>16 Bit Color(565)</td></tr><tr><td>10b</td><td>16 Bit Color(1555)</td></tr><tr><td>11b</td><td>32 Bit Color</td></tr></table></td></tr>
<tr><td>23:16</td><td>Raster Operation</td></tr>
<tr><td>15:0</td><td>Destination Pitch in DWords
2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).</td></tr>
<tr><td rowspan="2">2
BR22</td><td>31:16</td><td>Destination Y1 Coordinate (Top)
16 bit signed number.</td></tr>
<tr><td>15:0</td><td>Destination X1 Coordinate (Left)
16 bit signed number.</td></tr>
<tr><td rowspan="2">3
BR23</td><td>31:16</td><td>Destination Y2 Coordinate (Bottom)
16 bit signed number.</td></tr>
<tr><td>15:0</td><td>Destination X2 Coordinate (Right)
16 bit signed number.</td></tr>
<tr><td>4
BR09</td><td>31:0</td><td>Destination Base Address
<table><tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr></table></td></tr>
</table>

XY_MONO_PAT_BLT		
		Base address of the destination surface: X=0, Y=0. Lower 32bits of the 48bit addressing. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.
5 BR27	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.
	15:0	**Destination Base Address High** Format: GraphicsAddress[47:32] Should be programmed with the upper 16bits of the 48bit addressing.
6 BR16	31:0	**Pattern Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
7 BR17	31:0	**Pattern Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
8 BR20	31:0	**Pattern Data 0**
9 BR21	31:0	**Pattern Data 1**

XY_MONO_PAT_FIXED_BLT

<table>
<tr><th colspan="3">XY_MONO_PAT_FIXED_BLT</th></tr>
<tr><td colspan="3">Project: CHV, BSW
Source: BlitterCS
Length Bias: 2</td></tr>
<tr><td colspan="3">MONO_PAT_FIXED_BLT is used when we have no source and the monochrome pattern is not trivial (is not a solid color only). The monochrome pattern is one of 10 fixed patterns described below. The pattern seeds can still be used with the fixed patterns, creating even more fixed patterns. This eliminates 2 doublewords compared to the XY_MONO_PAT_BLT command packet.
All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.
The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.
The monochrome pattern transparency mode indicates whether to use the pattern background color or de-assert the write enables when the bit in the pattern is 0. When the pattern bit is 1, then the pattern foreground color is used in the ROP operation.</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="5">0

BR00</td><td>31:29</td><td>Client
<table><tr><td>Default Value:</td><td>02h 2D Processor</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td>28:22</td><td>Instruction Target(Opcode)
<table><tr><td>Default Value:</td><td>59h</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td>21:20</td><td>32bpp Byte Mask
This field is only used for 32bpp.
<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>[Default]</td></tr><tr><td>1xb</td><td>Write Alpha Channel</td></tr><tr><td>x1b</td><td>Write RGB Channel</td></tr></table></td></tr>
<tr><td>19</td><td>Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>18:15</td><td>Fixed Pattern
<table><tr><th>Value</th><th>Name</th></tr><tr><td>0000b</td><td>HS_HORIZONTAL</td></tr><tr><td>0001b</td><td>HS_VERTICAL</td></tr><tr><td>0010b</td><td>HS_FDIAGONAL</td></tr><tr><td>0011b</td><td>HS_BDIAGONAL</td></tr></table></td></tr>
</table>

XY_MONO_PAT_FIXED_BLT

0100b	HS_CROSS
0101b	HS_DIAGCROSS
0110b	Reserved
0111b	Reserved
1000b	Screen Door
1001b	SD Wide
1010b	Walking Bit (one)
1011b	Walking Zero
1100b	Reserved
1101b	Reserved
1110b	Reserved
1111b	Reserved

14:12 **Pattern Horizontal Seed**

Pixel of the scan line to start on corresponding to DST X=0.

11 **Tiling Enable**

Value	Name	Description
0b	Tiling Disabled (Linear Blit)	
1b	Tiling Enabled	[CHV, BSW] [CHV, BSW]: Tile-X or Tile-Y.

10:8 **Pattern Vertical Seed**

Scan line of the 8x8 pattern to start on corresponding to DST Y=0.

7:0 **DWord Length**

Value	Name
06h	

1 BR13

31 **Reserved**

Format:	MBZ

30 **Clipping Enabled**

Value	Name
0b	Disabled
1b	Enabled

29 **Reserved**

Format:	MBZ

28 **Mono Pattern Transparency Mode**

Value	Name
0	Use Background
1	Transparency Enabled

27:26 **Reserved**

XY_MONO_PAT_FIXED_BLT

	25:24	**Color Depth** <table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>8 Bit Color</td></tr><tr><td>01b</td><td>16 Bit Color(565)</td></tr><tr><td>10b</td><td>16 Bit Color(1555)</td></tr><tr><td>11b</td><td>32 Bit Color</td></tr></table>
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4 BR09	31:0	**Destination Base Address** <table><tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr></table>Base address of the destination surface: X=0, Y=0. Lower 32bits of the 48bit addressing. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.
5 BR27	31:16	**Reserved** <table><tr><td>Format:</td><td>MBZ</td></tr></table>Should be programmed all 0's for 48bit addressing.
	15:0	**Destination Base Address High** <table><tr><td>Format:</td><td>GraphicsAddress[47:32]</td></tr></table>Should be programmed with the upper 16bits of the 48bit addressing.
6 BR16	31:0	**Pattern Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
7 BR17	31:0	**Pattern Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]

XY_MONO_SRC_COPY_BLT

XY_MONO_SRC_COPY_BLT

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

This BLT instruction performs a monochrome source copy where the only operands involved is a monochrome source and destination. The source and destination operands cannot overlap therefore the X and Y directions are always forward.

All non-text monochrome sources are word aligned. At the end of a scan line of monochrome source, all bits until the next word boundary are ignored. The monochrome source data bit position field [2:0] indicates the bit position within the first byte of the scan line that should be used as the first source pixel which corresponds to the destination X1 coordinate.

The monochrome source transparency mode indicates whether to use the source background color or de-assert the write enables when the bit in the source is 0. When the source bit is 1, then the source foreground color is used in the ROP operation. The ROP value chosen must involve source and no pattern data in the ROP operation. Negative Stride (= Pitch) is NOT ALLOWED.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="7">0

BR00</td><td>31:29</td><td>Client<table><tr><td>Default Value:</td><td>02h 2D Processor</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td>28:22</td><td>Instruction Target(Opcode)<table><tr><td>Default Value:</td><td>54h</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td>21:20</td><td>32bpp Byte Mask
This field is only used for 32bpp.<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>[Default]</td></tr><tr><td>1xb</td><td>Write Alpha Channel</td></tr><tr><td>x1b</td><td>Write RGB Channel</td></tr></table></td></tr>
<tr><td>19:17</td><td>Monochrome source data bit position of the first pixel within a byte per scan line.</td></tr>
<tr><td>16:12</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>11</td><td>Tiling Enable<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0b</td><td>Tiling Disabled (Linear Blit)</td><td></td></tr><tr><td>1b</td><td>Tiling Enabled</td><td>[CHV, BSW] [CHV, BSW]: Tile-X or Tile-Y.</td></tr></table></td></tr>
<tr><td>10:8</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
</table>

<table>
<tr><th colspan="3">XY_MONO_SRC_COPY_BLT</th></tr>
<tr><td></td><td>7:0</td><td>DWord Length
<table><tr><th>Value</th><th>Name</th></tr><tr><td>08h</td><td></td></tr></table></td></tr>
<tr><td>1

BR13</td><td>31</td><td>Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>30</td><td>Clipping Enabled
<table><tr><th>Value</th><th>Name</th></tr><tr><td>0b</td><td>Disabled</td></tr><tr><td>1b</td><td>Enabled</td></tr></table></td></tr>
<tr><td></td><td>29</td><td>Mono Source Transparency Mode
<table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>Use Background</td></tr><tr><td>1</td><td>Transparency Enabled</td></tr></table></td></tr>
<tr><td></td><td>28:26</td><td>Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>25:24</td><td>Color Depth
<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>8 Bit Color</td></tr><tr><td>01b</td><td>16 Bit Color(565)</td></tr><tr><td>10b</td><td>16 Bit Color(1555)</td></tr><tr><td>11b</td><td>32 Bit Color</td></tr></table></td></tr>
<tr><td></td><td>23:16</td><td>Raster Operation</td></tr>
<tr><td></td><td>15:0</td><td>Destination Pitch in DWords
2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).</td></tr>
<tr><td>2

BR22</td><td>31:16</td><td>Destination Y1 Coordinate (Top)
16 bit signed number.</td></tr>
<tr><td></td><td>15:0</td><td>Destination X1 Coordinate (Left)
16 bit signed number.</td></tr>
<tr><td>3

BR23</td><td>31:16</td><td>Destination Y2 Coordinate (Bottom)
16 bit signed number.</td></tr>
<tr><td></td><td>15:0</td><td>Destination X2 Coordinate (Right)
16 bit signed number.</td></tr>
<tr><td>4

BR09</td><td>31:0</td><td>Destination Base Address
<table><tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr></table>Base address of the destination surface: X=0, Y=0. Lower 32bits of the 48bit addressing. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.</td></tr>
</table>

XY_MONO_SRC_COPY_BLT		
5 BR27	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.
	15:0	**Destination Base Address High** Format: GraphicsAddress[47:32] Should be programmed with the upper 16bits of the 48bit addressing.
6 BR12	31:0	**Mono Source Address** Format: GraphicsAddress[31:0] (address corresponds to DST X1, Y1) (Note no NPO2 change here). Lower 32bits of the 48bit addressing. This Monosource Base Address programmed, must always be Cache Line (64byte) aligned.
7 BR28	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.
	15:0	**Mono Source Address High** Format: GraphicsAddress[47:32] Should be programmed with the upper 16bits of the 48bit addressing.
8 BR18	31:0	**Source Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
9 BR19	31:0	**Source Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]

XY_MONO_SRC_COPY_IMMEDIATE_BLT

XY_MONO_SRC_COPY_IMMEDIATE_BLT

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

This instruction allows the Driver to send monochrome data through the instruction stream, eliminating the read latency of the source during command execution.
The IMMEDIATE_BLT data MUST transfer an even number of doublewords and the exact number of quadwords. DWL indicates the total number of Dwords of immediate data.
All non-text monochrome sources are word aligned. At the end of a scan line of monochrome source, all bits until the next word boundary are ignored. The Monochrome source data bit position field [2:0] indicates the bit position within the first byte of the scan line that should be used as the first source pixel which corresponds to the destination X1 coordinate.
The monochrome source transparency mode indicates whether to use the source background color or de-assert the write enables when the bit in the source is 0. When the source bit is 1, then the source foreground color is used in the ROP operation. The ROP value chosen must involve source and no pattern data in the ROP operation.
The monochrome source data supplied corresponds to the Destination X1 and Y1 coordinates.
Negative Stride (= Pitch) is NOT ALLOWED.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 71h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. Value / Name: 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:17	**Monochrome source data bit position of the first pixel within a byte per scan line.**
	16:12	**Reserved** Format: MBZ
	11	**Tiling Enable** Value / Name / Description: 0b Tiling Disabled (Linear) 1b Tiling Enabled [CHV, BSW] [CHV, BSW]: Tile-X or Tile-Y.

XY_MONO_SRC_COPY_IMMEDIATE_BLT		
	10:8	**Reserved**
	7:0	**DWord Length** Default Value: 06h Excludes DWORD 0,1 06 + DWL = (Number of Immediate double words)h
1 BR13	31	**Reserved**
	30	**Clipping Enabled** Value / Name: 0b: Disabled 1b: Enabled
	29	**Mono Source Transparency Mode** Value / Name: 0b: Transparency Enabled 1b: Use Background
	28:26	**Reserved**
	25:24	**Color Depth** Value / Name: 00b: 8 Bit Color 01b: 16 Bit Color(565) 10b: 16 Bit Color(1555) 11b: 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4 BR09	31:0	**Destination Base Address** Format: GraphicsAddress[31:0] Base address of the destination surface: X=0, Y=0. Lower 32bits of the 48bit addressing. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.

XY_MONO_SRC_COPY_IMMEDIATE_BLT		
5 BR27	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.
	15:0	**Destination Base Address High** Format: GraphicsAddress[47:32] Should be programmed with the upper 16bits of the 48bit addressing.
6 BR18	31:0	**Source Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
7 BR19	31:0	**Source Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
8..n	31:0	**Immediate Data**

XY_PAT_BLT

<table>
<tr><th colspan="3">XY_PAT_BLT</th></tr>
<tr><td colspan="3">Project: CHV, BSW
Source: BlitterCS
Length Bias: 2</td></tr>
<tr><td colspan="3">PAT_BLT is used when there is no source and the color pattern is not trivial (is not a solid color only).
If clipping is enabled, all scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.
The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="8">0

BR00</td><td>31:29</td><td>Client<table><tr><td>Default Value:</td><td>02h 2D Processor</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td>28:22</td><td>Instruction Target(Opcode)<table><tr><td>Default Value:</td><td>51h</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td>21:20</td><td>32bpp Byte Mask
This field is only used for 32bpp.<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>[Default]</td></tr><tr><td>1xb</td><td>Write Alpha Channel</td></tr><tr><td>x1b</td><td>Write RGB Channel</td></tr></table></td></tr>
<tr><td>19:15</td><td>Reserved</td></tr>
<tr><td>14:12</td><td>Pattern Horizontal Seed
Pixel of the scan line to start on corresponding to DST X=0.</td></tr>
<tr><td>11</td><td>Tiling Enable<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0b</td><td>Tiling Disabled (Linear Blit)</td><td></td></tr><tr><td>1b</td><td>Tiling Enabled</td><td>[CHV, BSW] [CHV, BSW]: Tile-X or Tile-Y.</td></tr></table></td></tr>
<tr><td>10:8</td><td>Pattern Vertical Seed
Scan line of the 8x8 pattern to start on corresponding to DST Y=0.</td></tr>
<tr><td>7:0</td><td>DWord Length<table><tr><td>Default Value:</td><td>06h</td></tr></table></td></tr>
<tr><td>1</td><td>31</td><td>Reserved</td></tr>
</table>

XY_PAT_BLT

BR13	30	**Clipping Enabled** <table><tr><th>Value</th><th>Name</th></tr><tr><td>0b</td><td>Disabled</td></tr><tr><td>1b</td><td>Enabled</td></tr></table>
	29:26	**Reserved** <table><tr><td>Format:</td><td>MBZ</td></tr></table>
	25:24	**Color Depth** <table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>8 Bit Color</td></tr><tr><td>01b</td><td>16 Bit Color(565)</td></tr><tr><td>10b</td><td>16 Bit Color(1555)</td></tr><tr><td>11b</td><td>32 Bit Color</td></tr></table>
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement (Negative Pitch Not allowed for Pixel nor Text) For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4 BR09	31:0	**Destination Base Address** <table><tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr></table>Base address of the destination surface: X=0, Y=0. Lower 32bits of the 48bit addressing. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.
5 BR27	31:16	**Reserved** <table><tr><td>Format:</td><td>MBZ</td></tr></table>Should be programmed all 0's for 48bit addressing.
	15:0	**Destination Base Address High** <table><tr><td>Format:</td><td>GraphicsAddress[47:32]</td></tr></table>Should be programmed with the upper 16bits of the 48bit addressing.

XY_PAT_BLT		
6 BR15	31:0	**Pattern Base Address** Format: GraphicsAddress[31:0] (28:06 are implemented) (Note no NPO2 change here). Lower 32bits of the 48bit addressing. The pattern data must be located in linear memory. The Pattern Base Address programmed, must always be Cache Line (64byte) aligned.
7 BR29	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.
	15:0	**Pattern Base Address High** Format: GraphicsAddress[47:32] Should be programmed with the upper 16bits of the 48bit addressing.

XY_PAT_BLT_IMMEDIATE

XY_PAT_BLT_IMMEDIATE

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

PAT_BLT_IMMEDIATE is used when there is no source and the color pattern is not trivial (is not a solid color only) and the pattern is pulled through the command stream. The immediate data sizes are 64 bytes (16 DWs), 128 bytes (32 DWs), or 256 (64DWs) for 8, 16, and 32 bpp color patterns.

DWL indicates the total number of Dwords of immediate data. All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.

The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="8">0

BR00</td><td>31:29</td><td>Client<table><tr><td>Default Value:</td><td>02h 2D Processor</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td>28:22</td><td>Instruction Target(Opcode)<table><tr><td>Default Value:</td><td>72h</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td>21:20</td><td>32bpp Byte Mask
This field is only used for 32bpp.<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>[Default]</td></tr><tr><td>1xb</td><td>Write Alpha Channel</td></tr><tr><td>x1b</td><td>Write RGB Channel</td></tr></table></td></tr>
<tr><td>19:15</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>14:12</td><td>Pattern Horizontal Seed
Pixel of the scan line to start on corresponding to DST X=0.</td></tr>
<tr><td>11</td><td>Tiling Enable<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0b</td><td>Tiling Disabled (Linear Blit)</td><td></td></tr><tr><td>1b</td><td>Tiling Enabled</td><td>[CHV, BSW] [CHV, BSW]: Tile-X or Tile-Y.</td></tr></table></td></tr>
<tr><td>10:8</td><td>Pattern Vertical Seed
Scan line of the 8x8 pattern to start on corresponding to DST Y=0.</td></tr>
<tr><td>7:0</td><td>DWord Length</td></tr>
</table>

XY_PAT_BLT_IMMEDIATE

DWord	Bit	Description
		Default Value: 04h Excludes DWORD 0,1 04 + DWL = (Number of Immediate double)h
1 BR13	31	**Reserved** Format: MBZ
	30	**Clipping Enabled** Value / Name: 0b: Disabled 1b: Enabled
	29:26	**Reserved** Format: MBZ
	25:24	**Color Depth** Value / Name: 00b: 8 Bit Color 01b: 16 Bit Color(565) 10b: 16 Bit Color(1555) 11b: 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement (Negative Pitch Not allowed for Pixel nor Text) For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4 BR09	31:0	**Destination Base Address** Format: GraphicsAddress[31:0] Base address of the destination surface: X=0, Y=0. Lower 32bits of the 48bit addressing. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.
5 BR27	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.

XY_PAT_BLT_IMMEDIATE		
	15:0	**Destination Base Address High** Format: GraphicsAddress[47:32] Should be programmed with the upper 16bits of the 48bit addressing.
6..n	31:0	**Immediate Data**

XY_PAT_CHROMA_BLT

<table>
<tr><th colspan="3">XY_PAT_CHROMA_BLT</th></tr>
<tr><td colspan="3">Project: CHV, BSW
Source: BlitterCS
Length Bias: 2</td></tr>
<tr><td colspan="3">PAT_BLT is used when there is no source and the color pattern is not trivial (is not a solid color only).
All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.
The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="9">0

BR00</td><td>31:29</td><td>Client
<table><tr><td>Default Value:</td><td>02h 2D Processor</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td>28:22</td><td>Instruction Target(Opcode)
<table><tr><td>Default Value:</td><td>76h</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td>21:20</td><td>32bpp Byte Mask
This field is only used for 32bpp.<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>[Default]</td></tr><tr><td>1xb</td><td>Write Alpha Channel</td></tr><tr><td>x1b</td><td>Write RGB Channel</td></tr></table></td></tr>
<tr><td>19:17</td><td>Transparency Range Mode
(chroma-key) - Dst Chroma-key modes ONLY (SRC ILLEGAL)</td></tr>
<tr><td>16:15</td><td>Reserved</td></tr>
<tr><td>14:12</td><td>Pattern Horizontal Seed
Pixel of the scan line to start on corresponding to DST X=0.</td></tr>
<tr><td>11</td><td>Tiling Enable<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0b</td><td>Tiling Disabled (Linear Blit)</td><td></td></tr><tr><td>1b</td><td>Tiling Enabled</td><td>[CHV, BSW] [CHV, BSW]: Tile-X or Tile-Y.</td></tr></table></td></tr>
<tr><td>10:8</td><td>Pattern Vertical Seed
Scan line of the 8x8 pattern to start on corresponding to DST Y=0.</td></tr>
<tr><td>7:0</td><td>DWord Length<table><tr><td>Default Value:</td><td>08h Excludes DWORD 0,1</td></tr></table></td></tr>
</table>

XY_PAT_CHROMA_BLT

DWord	Bit	Description
1 BR13	31	**Reserved** Format: MBZ
	30	**Clipping Enabled** Value | Name 0b | Disabled 1b | Enabled
	29:26	**Reserved** Format: MBZ
	25:24	**Color Depth** Value | Name 00b | 8 Bit Color 01b | 16 Bit Color(565) 10b | 16 Bit Color(1555) 11b | 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement (Negative Pitch Not allowed for Pixel nor Text) For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4 BR09	31:0	**Destination Base Address** Format: GraphicsAddress[31:0] Base address of the destination surface: X=0, Y=0. Lower 32bits of the 48bit addressing. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.
5 BR27	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.
	15:0	**Destination Base Address High** Format: GraphicsAddress[47:32] Should be programmed with the upper 16bits of the 48bit addressing.

XY_PAT_CHROMA_BLT		
6 BR15	31:0	**Pattern Base Address** Format: GraphicsAddress[31:0] (28:06 are implemented) (Note no NPO2 change here). Lower 32bits of the 48bit addressing. The pattern data must be located in linear memory. The Pattern Base Address programmed, must always be Cache Line (64byte) aligned.
7 BR29	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.
	15:0	**Pattern Base Address High** Format: GraphicsAddress[47:32] Should be programmed with the upper 16bits of the 48bit addressing.
8 BR18	31:0	**Transparency Color Low** (Chroma-key Low = Pixel Greater or Equal)
9 BR19	31:0	**Transparency Color High** (Chroma-key High = Pixel Less or Equal)

XY_PAT_CHROMA_BLT_IMMEDIATE

<table>
<tr><th colspan="3">XY_PAT_CHROMA_BLT_IMMEDIATE</th></tr>
<tr><td colspan="3">Project: CHV, BSW
Source: BlitterCS
Length Bias: 2</td></tr>
<tr><td colspan="3">PAT_BLT_IMMEDIATE is used when there is no source and the color pattern is not trivial (is not a solid color only) and the pattern is pulled through the command stream. The immediate data sizes are 64 bytes (16 DWs), 128 bytes (32 DWs), or 256 (64DWs) for 8, 16, and 32 bpp color patterns.
DWL indicates the total number of Dwords of immediate data. All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.
The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="8">0

BR00</td><td>31:29</td><td>Client<table><tr><td>Default Value:</td><td>02h 2D Processor</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td>28:22</td><td>Instruction Target(Opcode)<table><tr><td>Default Value:</td><td>77h</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td>21:20</td><td>32bpp Byte Mask
This field is only used for 32bpp.<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>[Default]</td></tr><tr><td>1xb</td><td>Write Alpha Channel</td></tr><tr><td>x1b</td><td>Write RGB Channel</td></tr></table></td></tr>
<tr><td>19:17</td><td>Transparency Range Mode
(chroma-key) - Dst Chroma-key modes ONLY (SRC ILLEGAL)</td></tr>
<tr><td>16:15</td><td>Reserved</td></tr>
<tr><td>14:12</td><td>Pattern Horizontal Seed
Pixel of the scan line to start on corresponding to DST X=0.</td></tr>
<tr><td>11</td><td>Tiling Enable<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0b</td><td>Tiling Disabled (Linear Blit)</td><td></td></tr><tr><td>1b</td><td>Tiling Enabled</td><td>[CHV, BSW] [CHV, BSW]: Tile-X or Tile-Y.</td></tr></table></td></tr>
<tr><td>10:8</td><td>Pattern Vertical Seed
Scan line of the 8x8 pattern to start on corresponding to DST Y=0.</td></tr>
</table>

XY_PAT_CHROMA_BLT_IMMEDIATE		
	7:0	**DWord Length** Default Value: 06h Excludes DWORD 0,1 06 + DWL = (Number of Immediate double)h
1 BR13	31	**Reserved** Format: MBZ
	30	**Clipping Enabled** Value / Name: 0b: Disabled 1b: Enabled
	29:26	**Reserved** Format: MBZ
	25:24	**Color Depth** Value / Name: 00b: 8 Bit Color 01b: 16 Bit Color(565) 10b: 16 Bit Color(1555) 11b: 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4 BR09	31:0	**Destination Base Address** Format: GraphicsAddress[31:0] Base address of the destination surface: X=0, Y=0. Lower 32bits of the 48bit addressing. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.
5 BR27	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.

XY_PAT_CHROMA_BLT_IMMEDIATE		
	15:0	**Destination Base Address High** Format: GraphicsAddress[47:32] Should be programmed with the upper 16bits of the 48bit addressing.
6 BR18	31:0	**Transparency Color Low** (Chroma-key Low = Pixel Greater or Equal)
7 BR19	31:0	**Transparency Color High** (Chroma-key High = Pixel Less or Equal)
8..n	31:0	**Immediate Data**

XY_PIXEL_BLT

XY_PIXEL_BLT

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

The Destination X coordinate and Destination Y coordinate is compared with the ClipRect registers. If it is within all 4 comparisons, then the pixel supplied in the XY_SETUP_BLT instruction is written with the raster operation to (Destination Y Address + (Destination Y coordinate * Destination pitch) + (Destination X coordinate * bytes per pixel)).
ROP field must specify pattern or fill with 0's or 1's. There is no source operand.
Negative Stride (= Pitch) specified in the Setup command is Not Allowed

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 24h Format: Opcode
	21:12	**Reserved** Format: MBZ
	11	**Tiling Enable** Value / Name / Description: 0b / Tiling Disabled (Linear Blit) / 1b / Tiling Enabled / [CHV, BSW] [CHV, BSW]: Tile-X or Tile-Y.
	10:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 00h
1 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.

XY_SCANLINES_BLT

XY_SCANLINES_BLT		
Project:	CHV, BSW	
Source:	BlitterCS	
Length Bias:	2	

All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.
The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.
Solid pattern should use the XY_SETUP_MONO_PATTERN_SL_BLT instruction.
ROP field must specify pattern or fill with 0's or 1's. There is no source operand.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 25h Format: Opcode
	21:15	**Reserved**
	14:12	**Pattern Horizontal Seed** Pixel of the scan line to start on corresponding to DST X=0.
	11	**Tiling Enable** Value / Name / Description: 0b — Tiling Disabled (Linear Blit) 1b — Tiling Enabled — [CHV, BSW] [CHV, BSW]: Tile-X or Tile-Y.
	10:8	**Pattern Vertical Seed** Scan line of the 8x8 pattern to start on corresponding to DST Y=0.
	7:0	**DWord Length** Default Value: 01h
1 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
2 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.

XY_SETUP_BLT

XY_SETUP_BLT

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

This setup instruction supplies common setup information including clipping coordinates used by the XY commands: XY_PIXEL_BLT, XY_SCANLINE_BLT, XY_TEXT_BLT, and XY_TEXT_BLT_IMMEDIATE.
These are the only instructions that require that state be saved between instructions other than the Clipping parameters. There are 5 dedicated registers to contain the state for the 3 setup BLT instructions (XY_SETUP_BLT, XY_SETUP_MONO_PATTERN_SL_BLT, and XY_SETUP_CLIP_BLT. All other BLTs use a temporary version of these. The 5 double word registers are: DW1 (Setup Control), DW6 (Setup Foreground color), DW5 (Setup Background color), DW7 (Setup Pattern address), and DW4 (Setup Destination Base Address).

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 01h Format: Opcode
	21:20	**32 bpp Byte Mask** Value / Name: 1xb: Write Alpha Channel x1b: Write RGB Channel
	19:12	**Reserved**
	11	**Tiling Enable** Value / Name: 0b: Tiling Disabled (Linear Blit) 1b: Tiling Enabled (Tile-X or Tile-Y)
	10:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 08h
1 BR01	31	**Reserved**
	30	**Clipping Enabled** Value / Name: 0b: Disabled 1b: Enabled

<table>
<tr><th colspan="3">XY_SETUP_BLT</th></tr>
<tr><td rowspan="5"></td><td>29</td><td>Mono Source Transparency Mode
<table><tr><th>Value</th><th>Name</th></tr><tr><td>0b</td><td>Use Background</td></tr><tr><td>1b</td><td>Transparency Enabled</td></tr></table></td></tr>
<tr><td>28:26</td><td>Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>25:24</td><td>Color Depth
<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>8 Bit Color</td></tr><tr><td>01b</td><td>16 Bit Color(565)</td></tr><tr><td>10b</td><td>16 Bit Color(1555)</td></tr><tr><td>11b</td><td>32 Bit Color</td></tr></table></td></tr>
<tr><td>23:16</td><td>Raster Operation</td></tr>
<tr><td>15:0</td><td>Destination Pitch in DWords
2's complement (Negative Pitch Not allowed for Pixel nor Text) For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).</td></tr>
<tr><td rowspan="2">2
BR24</td><td>31:16</td><td>ClipRect Y1 Coordinate (Top)
(30:16 = 15 bit positive number)</td></tr>
<tr><td>15:0</td><td>ClipRect X1 Coordinate (Left)
(14:00 = 15 bit positive number)</td></tr>
<tr><td rowspan="2">3
BR25</td><td>31:16</td><td>ClipRect Y2 Coordinate (Bottom)
(30:16 = 15 bit positive number)</td></tr>
<tr><td>15:0</td><td>ClipRect X2 Coordinate (Right)
(14:00 = 15 bit positive number)</td></tr>
<tr><td>4
BR09</td><td>31:0</td><td>Setup Destination Base Address
<table><tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr></table>
Base address of the destination surface: X=0, Y=0. Lower 32bits of the 48bit addressing. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.</td></tr>
<tr><td rowspan="2">5
BR27</td><td>31:16</td><td>Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table>
Should be programmed all 0's for 48bit addressing.</td></tr>
<tr><td>15:0</td><td>Setup Destination Base Address High
<table><tr><td>Format:</td><td>GraphicsAddress[47:32]</td></tr></table>
Should be programmed with the upper 16bits of the 48bit addressing.</td></tr>
</table>

XY_SETUP_BLT		
6 BR05	31:0	**Setup Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0] All
7 BR06	31:0	**Setup Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0] (SLB and TB only)
8 BR07	31:0	**Setup Pattern Base Address for Color Pattern** Format: GraphicsAddress[31:0] (26:06 are implemented) (SLB only) (Note no NPO2 change here). The pattern data must be located in linear memory. Lower 32bits of the 48bit addressing. The Pattern Base Address programmed, must always be Cache Line (64byte) aligned.
9 BR30	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.
	15:0	**Setup Pattern Base Address for Color Pattern High** Format: GraphicsAddress[47:32] Should be programmed with the upper 16bits of the 48bit addressing.

XY_SETUP_CLIP_BLT

XY_SETUP_CLIP_BLT

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

This command is used to only change the clip coordinate registers. These are the same clipping registers as the Setup clipping registers above.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 03h Format: Opcode
	21:12	**Reserved** Format: MBZ
	11	**Tiling Enable** Value / Name: 0b: Tiling Disabled (Linear Blit) 1b: Tiling Enabled (Tile-X or Tile-Y
	10:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 01h
1 BR24	31:16	**ClipRect Y1 Coordinate (Top)** (30:16 = 15 bit positive number)
	15:0	**ClipRect X1 Coordinate (Left)** (14:00 = 15 bit positive number)
2 BR25	31:16	**ClipRect Y2 Coordinate (Bottom)** (30:16 = 15 bit positive number)
	15:0	**ClipRect X2 Coordinate (Right)** (14:00 = 15 bit positive number)

XY_SETUP_MONO_PATTERN_SL_BLT

XY_SETUP_MONO_PATTERN_SL_BLT

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

This setup instruction supplies common setup information including clipping coordinates used exclusively with the following instruction: XY_SCANLINE_BLT (SLB) - 1 scan line of monochrome pattern and destination are the only operands allowed.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 11h Format: Opcode
	21:20	**32 bpp Byte Mask** Value / Name: 1xb: Write Alpha Channel x1b: Write RGB Channel
	19:12	**Reserved**
	11	**Tiling Enable** Value / Name: 0b: Tiling Disabled (Linear Blit) 1b: Tiling Enabled (Tile-X or Tile-Y
	10:8	**Reserved**
	7:0	**DWord Length** Default Value: 08h
1 BR01	31	**Solid Pattern Select** (SLB and Pixel only) Value / Name: 0: No Solid Pattern 1: Solid Pattern
	30	**Clipping Enabled** Value / Name: 0b: Disabled 1b: Enabled

XY_SETUP_MONO_PATTERN_SL_BLT

<table>
<tr><td></td><td>29</td><td>Reserved
Format: | MBZ</td></tr>
<tr><td></td><td>28</td><td>Mono Pattern Transparency Mode
<table><tr><th>Value</th><th>Name</th></tr><tr><td>0b</td><td>Use Background</td></tr><tr><td>1b</td><td>Transparency Enabled</td></tr></table></td></tr>
<tr><td></td><td>27:26</td><td>Reserved
Format: | MBZ</td></tr>
<tr><td></td><td>25:24</td><td>Color Depth
<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>8 Bit Color</td></tr><tr><td>01b</td><td>16 Bit Color(565)</td></tr><tr><td>10b</td><td>16 Bit Color(1555)</td></tr><tr><td>11b</td><td>32 Bit Color</td></tr></table></td></tr>
<tr><td></td><td>23:16</td><td>Raster Operation</td></tr>
<tr><td></td><td>15:0</td><td>Destination Pitch in DWords
2's complement (Negative Pitch Not allowed for Pixel nor Text) For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).</td></tr>
<tr><td>2
BR24</td><td>31:16</td><td>ClipRect Y1 Coordinate (Top)
(30:16 = 15 bit positive number)</td></tr>
<tr><td></td><td>15:0</td><td>ClipRect X1 Coordinate (Left)
(14:00 = 15 bit positive number)</td></tr>
<tr><td>3
BR25</td><td>31:16</td><td>ClipRect Y2 Coordinate (Bottom)
(30:16 = 15 bit positive number)</td></tr>
<tr><td></td><td>15:0</td><td>ClipRect X2 Coordinate (Right)
(14:00 = 15 bit positive number)</td></tr>
<tr><td>4
BR09</td><td>31:0</td><td>Setup Destination Base Address
Format: | GraphicsAddress[31:0]
Base address of the destination surface: X=0, Y=0. Lower 32bits of the 48bit addressing. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.</td></tr>
<tr><td>5
BR27</td><td>31:16</td><td>Reserved
Format: | MBZ
Should be programmed all 0's for 48bit addressing.</td></tr>
<tr><td></td><td>15:0</td><td>Setup Destination Base Address High
Format: | GraphicsAddress[47:32]
Should be programmed with the upper 16bits of the 48bit addressing.</td></tr>
</table>

XY_SETUP_MONO_PATTERN_SL_BLT		
6 BR05	31:0	**Setup Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0] All
7 BR06	31:0	**Setup Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0] (SLB and TB only)
8 BR20	31:0	**DW0 (least significant) for a Monochrome Pattern**
9 BR21	31:0	**DW1 (most significant) for a Monochrome Pattern**

XY_SRC_COPY_BLT

XY_SRC_COPY_BLT

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

This BLT instruction performs a color source copy where the only operands involved is a color source and destination of the same bit width.
The source and destination operands may overlap, which means that the X and Y directions can be either forward or backwards. The BLT Engine takes care of all situations. The base addresses plus the X and Y coordinates determine if there is an overlap between the source and destination operands. If the base addresses of the source and destination are the same and the Source X1 is less than Destination X1, then the BLT Engine performs the accesses in the X-backwards access pattern. There is no need to look for an actual overlap. If the base addresses are the same and Source Y1 is less than Destination Y1, then the scan line accesses start at Destination Y2 with the corresponding source scan line and the strides are subtracted for every scan line access.
The ROP value chosen must involve source and no pattern data in the ROP operation.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="7">0

BR00</td><td>31:29</td><td>Client
<table><tr><td>Default Value:</td><td>02h 2D Processor</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td>28:22</td><td>Instruction Target(Opcode)
<table><tr><td>Default Value:</td><td>53h</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td>21:20</td><td>32bpp Byte Mask
This field is only used for 32bpp.
<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>[Default]</td></tr><tr><td>1xb</td><td>Write Alpha Channel</td></tr><tr><td>x1b</td><td>Write RGB Channel</td></tr></table></td></tr>
<tr><td>19:16</td><td>Reserved</td></tr>
<tr><td>15</td><td>Src Tiling Enable
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0b</td><td>Tiling Disabled (Linear)</td><td></td></tr><tr><td>1b</td><td>Tiling Enabled</td><td>[CHV, BSW] [CHV, BSW]: Tile-X or Tile-Y.</td></tr></table></td></tr>
<tr><td>14:12</td><td>Reserved</td></tr>
<tr><td>11</td><td>Dest Tiling Enable
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0b</td><td>Tiling Disabled (Linear Blit)</td><td></td></tr><tr><td>1b</td><td>Tiling Enabled</td><td>[CHV, BSW] [CHV, BSW]: Tile-X or Tile-Y.</td></tr></table></td></tr>
</table>

XY_SRC_COPY_BLT

DWord	Bit	Description
	10:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Value: 08h — Name: (none)
1 BR13	31	**Reserved** Format: MBZ
	30	**Clipping Enabled** 0b: Disabled 1b: Enabled
	29:26	**Reserved** Format: MBZ
	25:24	**Color Depth** 00b: 8 Bit Color 01b: 16 Bit Color(565) 10b: 16 Bit Color(1555) 11b: 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4 BR09	31:0	**Destination Base Address** Format: GraphicsAddress[31:0] Base address of the destination surface: X=0, Y=0. Lower 32bits of the 48bit addressing. When Tiling is enabled (Bit_11 enabled), this address must be 4KB-aligned. When Tiling is not enabled, this address should be CL (64byte) aligned.

XY_SRC_COPY_BLT		
5 BR27	31:16	**Reserved** Format: MBZ Must be all 0's for 48bit addressing.
	15:0	**Destination Base Address High** Format: GraphicsAddress[47:32] The upper 16bits of the 48-bit address.
6 BR26	31:16	**Source Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Source X1 Coordinate (Left)** 16 bit signed number.
7 BR11	31:16	**Reserved** Format: MBZ
	15:0	**Source Pitch (double word aligned) and in DWords** 2's complement. For Tiled Src (bit 15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Yand can be upto 128Kbytes (or 32KDwords).
8 BR12	31:0	**Source Base Address** Format: GraphicsAddress[31:0] Base address of the destination surface: X=0, Y=0. Lower 32bits of the 48bit addressing. When Src Tiling is enabled (Bit_15 enabled), this address must be 4KB-aligned. When Tiling is not enabled, this address should be CL (64byte) aligned.
9 BR28	31:16	**Reserved** Format: MBZ Must be all 0's for 48-bit addressing.
	15:0	**Source Base Address High** Format: GraphicsAddress[47:32] The upper 16 bits of the 48-bit address.

XY_SRC_COPY_CHROMA_BLT

XY_SRC_COPY_CHROMA_BLT

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

This BLT instruction performs a color source copy with chroma-keying where the only operands involved is a color source and destination of the same bit width.
The source and destination operands may overlap, which means that the X and Y directions can be either forward or backwards. The BLT Engine takes care of all situations. The base addresses plus the X and Y coordinates determine if there is an overlap between the source and destination operands. If the base addresses of the source and destination are the same and the Source X1 is less than Destination X1, then the BLT Engine performs the accesses in the X-backwards access pattern. There is no need to look for an actual overlap. If the base addresses are the same and Source Y1 is less than Destination Y1, then the scan line accesses start at Destination Y2 with the corresponding source scan line and the strides are subtracted for every scan line access.
The ROP value chosen must involve source and no pattern data in the ROP operation.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="7">0

BR00</td><td>31:29</td><td>Client<table><tr><td>Default Value:</td><td>02h 2D Processor</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td>28:22</td><td>Instruction Target(Opcode)<table><tr><td>Default Value:</td><td>73h</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td>21:20</td><td>32bpp Byte Mask
This field is only used for 32bpp.<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>[Default]</td></tr><tr><td>1xb</td><td>Write Alpha Channel</td></tr><tr><td>x1b</td><td>Write RGB Channel</td></tr></table></td></tr>
<tr><td>19:17</td><td>Transparency Range Mode
(chroma-key)</td></tr>
<tr><td>16</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>15</td><td>Src Tiling Enable<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0b</td><td>Tiling Disabled (Linear)</td><td></td></tr><tr><td>1b</td><td>Tiling Enabled</td><td>[CHV, BSW] [CHV, BSW]: Tile-X or Tile-Y.</td></tr></table></td></tr>
<tr><td>14:12</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
</table>

XY_SRC_COPY_CHROMA_BLT

DWord	Bit	Description
	11	**Dest Tiling Enable** <table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0b</td><td>Tiling Disabled (Linear Blit)</td><td></td></tr><tr><td>1b</td><td>Tiling Enabled</td><td>[CHV, BSW] [CHV, BSW]: Tile-X or Tile-Y.</td></tr></table>
	10:8	**Reserved** Format: MBZ
	7:0	**DWord Length** <table><tr><th>Value</th><th>Name</th></tr><tr><td>0Ah</td><td></td></tr></table>
1 BR13	31	**Reserved** Format: MBZ
	30	**Clipping Enabled** <table><tr><th>Value</th><th>Name</th></tr><tr><td>0b</td><td>Disabled</td></tr><tr><td>1b</td><td>Enabled</td></tr></table>
	29:26	**Reserved** Format: MBZ
	25:24	**Color Depth** <table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>8 Bit Color</td></tr><tr><td>01b</td><td>16 Bit Color(565)</td></tr><tr><td>10b</td><td>16 Bit Color(1555)</td></tr><tr><td>11b</td><td>32 Bit Color</td></tr></table>
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4 BR09	31:0	**Destination Base Address** Format: GraphicsAddress[31:0]

XY_SRC_COPY_CHROMA_BLT		
		Base address of the destination surface: X=0, Y=0. Lower 32bits of the 48bit addressing. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.
5 BR27	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.
	15:0	**Destination Base Address High** Format: GraphicsAddress[47:32] Should be programmed with the upper 16bits of the 48bit addressing.
6 BR26	31:16	**Source Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Source X1 Coordinate (Left)** 16 bit signed number.
7 BR11	31:16	**Reserved** Format: MBZ
	15:0	**Source Pitch (double word aligned) and in DWords** 2's complement. For Tiled Src (bit 15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Yand can be upto 128Kbytes (or 32KDwords).
8 BR12	31:0	**Source Base Address** Format: GraphicsAddress[31:0] Base address of the destination surface: X=0, Y=0. Lower 32bits of the 48bit addressing. When Tiling is enabled (Bit_15 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.
9 BR28	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.
	15:0	**Source Base Address High** Format: GraphicsAddress[47:32] Should be programmed with the upper 16bits of the 48bit addressing.
10 BR18	31:0	**Transparency Color Low** (Chroma-key Low = Pixel Greater or Equal)
11 BR19	31:0	**Transparency Color High** (Chroma-key High = Pixel Less or Equal)

XY_TEXT_BLT

XY_TEXT_BLT		
Project: CHV, BSW Source: BlitterCS Length Bias: 2		
All source scan lines and pixels that fall within the ClipRect Y and X coordinates are written. The source address corresponds to Destination X1 and Y1 coordinate. Text is either bit or byte packed. Bit packed means that the next scan line starts 1 pixel after the end of the current scan line with no bit padding. Byte packed means that the next scan line starts on the first bit of the next byte boundary after the last bit of the current line. Source expansion color registers are always in the SETUP_BLT. Negative Stride (= Pitch) is NOT ALLOWED.		
DWord	**Bit**	**Description**
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 26h Format: Opcode
	21:17	**Reserved** Format: MBZ
	16	**Bit / Byte Packed** Byte packed is for the NT driver. Value / Name: 0 — **Bit** 1 — Byte
	15:12	**Reserved**
	11	**Tiling Enable** Value / Name / Description: 0b — Tiling Disabled (Linear Blit) 1b — Tiling Enabled — [CHV, BSW] [CHV, BSW]: Tile-X or Tile-Y.
	10:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 03h
1	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.

XY_TEXT_BLT		
BR22	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
2 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
3 BR12	31:0	**Source Address** Format: GraphicsAddress[31:0] (address of the first byte on scan line corresponding to Dst X1, Y1). Lower 32bits of the 48bit addressing. (Note no NPO2 change here). Since Text data is Monosource data, the Text source Base Address programmed, must always be Cache Line (64byte) aligned.
4 BR28	31:16	**Reserved** Format: MBZ Should be programmed with all "0"s for 48bit addressing.
	15:0	**Source Address High** Format: GraphicsAddress[47:32] Should be programmed with the upper 16bits of the 48bit addressing.

XY_TEXT_IMMEDIATE_BLT

XY_TEXT_IMMEDIATE_BLT

Project:	CHV, BSW
Source:	BlitterCS
Length Bias:	2

This instruction allows the Driver to send data through the instruction stream that eliminates the read latency of reading a source from memory.

If an operand is in system cacheable memory and either small or only accessed once, it can be copied directly to the instruction stream versus to graphics accessible memory. The IMMEDIATE_BLT data MUST transfer an even number of doublewords.

The BLT engine will hang if it does not get an even number of doublewords. All source scan lines and pixels that fall within the ClipRect X and Y coordinates are written. The source data corresponds to Destination X1 and Y1 coordinate.

Source expansion color registers are always in the SETUP_BLT. NEGATIVE STRIDE (= PITCH) IS NOT ALLOWED.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 31h Format: Opcode
	21:17	**Reserved**
	16	**Bit / Byte Packed** Byte packed is for the NT driver. Value 0 = Name **Bit** Value 1 = Name Byte
	15:12	**Reserved** Format: MBZ
	11	**Tiling Enable** 0b: Tiling Disabled (Linear Blit) 1b: Tiling Enabled — [CHV, BSW] [CHV, BSW]: Tile-X or Tile-Y.
	10:8	**Reserved**
	7:0	**DWord Length** Default Value: 01h Excludes DWORD 0,1 01 + DWL = (Number of Immediate double words)h

XY_TEXT_IMMEDIATE_BLT		
1 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
2 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
3..n	31:0	**Immediate Data**