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

Volume 2, Part 2: Command Reference - Instructions

For the 2014 Intel Atom™ Processors, Celeron™ Processors, and Pentium™ Processors based on the "BayTrail" Platform (ValleyView graphics)

Creative Commons License

Notices and Disclaimers

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Implementations of the I2C bus/protocol may require licenses from various entities, including Philips Electronics N.V. and North American Philips Corporation.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others.

Table of Contents

3DPRIMITIVE

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).
Workloads enabling topology filter using MI_TOPOLOGY_FILTER must always program PIPECONTROL command with only Post-Sync Operation (Write Immediate data) prior to every 3DPRIMTIVE command programmed.

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
	15:11	**Reserved** Format: MBZ
	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) • 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 must not be ENABLED at the same time, or behavior is UNDEFINED.

3DPRIMITIVE

	9	**Reserved** Format: MBZ
	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) Format: =n Total Length - 2
1	31:10	**Reserved** Format: MBZ
	9	**End Offset Enable** Format: Enable **Description** 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 One added restriction applies: • Instance Count = 1 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). 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.

<table>
<tr><th colspan="3">3DPRIMITIVE</th></tr>
<tr><td></td><td>8</td><td><b>Vertex Access Type</b>
This field specifies how data held in vertex buffers marked as VERTEXDATA is accessed by Vertex Fetch.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>SEQUENTIAL</td><td>VERTEXDATA buffers are accessed sequentiallyRequiref if End Offset Enable is ENABLED.</td></tr>
<tr><td>1h</td><td>RANDOM</td><td>VERTEXDATA buffers are accessed randomly via an index obtained from the Index Buffer.</td></tr>
</table></td></tr>
<tr><td></td><td>7:6</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>5:0</td><td><b>Primitive Topology Type</b>
<table>
<tr><td>Format:</td><td>3D_PrimTopoType See table below for encoding, see 3D Overview for diagrams and general comments</td></tr>
</table>
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.).</td></tr>
<tr><td>2</td><td>31:0</td><td><b>Vertex Count Per Instance</b>
<table>
<tr><td>Format:</td><td>U32 Count of vertices</td></tr>
</table>
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.
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>• 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'.</td></tr>
</table></td></tr>
</table>

<table>
<tr><th colspan="3">3DPRIMITIVE</th></tr>
<tr><td>3</td><td>31:0</td><td><b>Start Vertex Location</b>
<table><tr><td>Format:</td><td>U32 structure index</td></tr></table>
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>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><b>Instance Count</b>
<table><tr><td>Format:</td><td>U32 Count of instances</td></tr></table>
<table><tr><th>Description</th></tr>
<tr><td>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.</td></tr>
<tr><td>Must be set to 1 if End Offset Enable is ENABLED.</td></tr></table>
<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><b>Start Instance Location</b>
<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 INSTANCEDATA buffers.</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>
</table>

<table>
<tr><th colspan="3">3DPRIMITIVE</th></tr>
<tr><td>6</td><td>31:0</td><td><b>Base Vertex Location</b>
<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><ul>
<li>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).</li>
<li>Must be set to 0 if End Offset Enable is ENABLED.</li>
<li>Ignored if Indirect Parameter Enable is ENABLED.</li>
</ul></td></tr></table>
</td></tr>
</table>

3DSTATE_AA_LINE_PARAMETERS

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** Format: MBZ
	7:0	**Dword Length** Default Value: 1h Excludes Dword (0,1) Format: =n Total Length - 2
1	31:24	**Reserved** Format: MBZ
	23:16	**AA Coverage Bias** Format: U0.8 This field specifies the bias term to be used in the aa coverage computation for edges 0 and 3.
	15:8	**Reserved** Format: MBZ
	7:0	**AA Coverage Slope** 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	**Reserved** Format: MBZ

3DSTATE_AA_LINE_PARAMETERS

	23:16	**AA Coverage EndCap Bias** Format: U0.8 This field specifies the bias term to be used in the aa coverage computation for edges 1 and 2.
	15:8	**Reserved** Format: MBZ
	7:0	**AA Coverage EndCap Slope** 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_POINTERS_DS

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** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:16	**Reserved** Format: MBZ
	15:5	**Pointer to DS Binding Table** Format: SurfaceStateOffset[15:5]BINDING_TABLE_STATE*256 Specifies an aligned address offset of the function's BINDING_TABLE_STATE. This offset is relative to the **Surface State Base Address** in units of 32B.
	4:0	**Reserved** Format: MBZ

3DSTATE_BINDING_TABLE_POINTERS_GS

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** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:16	**Reserved** Format: MBZ
	15:5	**Pointer to GS Binding Table** Format: SurfaceStateOffset[15:5]BINDING_TABLE_STATE*256 Specifies an aligned address offset of the function's BINDING_TABLE_STATE. This offset is relative to the **Surface State Base Address** in units of 32B.
	4:0	**Reserved** Format: MBZ

3DSTATE_BINDING_TABLE_POINTERS_HS

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** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:16	**Reserved** Format: MBZ
	15:5	**Pointer to HS Binding Table** Format: SurfaceStateOffset[15:5]BINDING_TABLE_STATE*256 Specifies an aligned address offset of the function's BINDING_TABLE_STATE. This offset is relative to the **Surface State Base Address** in units of 32B.
	4:0	**Reserved** Format: MBZ

3DSTATE_BINDING_TABLE_POINTERS_PS

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** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:16	**Reserved** Format: MBZ
	15:5	**Pointer to PS Binding Table** Format: SurfaceStateOffset[15:5]BINDING_TABLE_STATE*256 Specifies an aligned address offset of the function's BINDING_TABLE_STATE. This offset is relative to the **Surface State Base Address** in units of 32B.
	4:0	**Reserved** Format: MBZ

3DSTATE_BINDING_TABLE_POINTERS_VS

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** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:16	**Reserved** Format: MBZ
	15:5	**Pointer to VS Binding Table** Format: SurfaceStateOffset[15:5]BINDING_TABLE_STATE*256 Specifies an aligned address offset of the function's BINDING_TABLE_STATE. This offset is relative to the **Surface State Base Address** in units of 32B.
	4:0	**Reserved** Format: MBZ

3DSTATE_BLEND_STATE_POINTERS

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** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:6	**Blend State Pointer** 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** Format: MBZ
	0	**Reserved** Format: MB0

3DSTATE_CC_STATE_POINTERS

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** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:6	**Color Calc State Pointer** 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** Format: MBZ
	0	**Reserved** Format: MB0

3DSTATE_CHROMA_KEY

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** Format: MBZ
	7:0	**DWord Length** Default Value: 2h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:30	**ChromaKey Table Index** Format: U2 index Selects which entry in the ChromaKey table is to be loaded Value: [0,3] Name:
	29:0	**Reserved** 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

3DSTATE_CHROMA_KEY

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

Source:	RenderCS
Length Bias:	2

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).

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).
3DSTATE_CLEAR_PARAMS must always be programmed in the along with the other Depth/Stencil state commands(i.e. 3DSTATE_DEPTH_BUFFER, 3DSTATE_STENCIL_BUFFER, or 3DSTATE_HIER_DEPTH_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 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** Format: for Surface Format of depth buffer:D32_FLOAT_S8X24_UINT: IEEE_FloatD32_FLOAT: IEEE_FloatD24_UNORM_S8_UINT: U24 UNORM in bits [23:0]D24_UNORM_X8_UINT: U24 UNORM in bits [23:0]D16_UNORM: U16 UNORM in bits [15:0] 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.

3DSTATE_CLEAR_PARAMS		
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

<table>
<tr><td colspan="3">Source: RenderCS
Length Bias: 2</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h GFXPIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Command SubType</b><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></td><td>26:24</td><td><b>3D Command Opcode</b><table><tr><td>Default Value:</td><td>0h 3DSTATE_PIPELINE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>23:16</td><td><b>3D Command Sub Opcode</b><table><tr><td>Default Value:</td><td>12h 3DSTATE_CLIP</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>02h Excludes DWord (0,1)</td></tr><tr><td>Format:</td><td>=n Total Length - 2</td></tr></table></td></tr>
<tr><td>1</td><td>31:21</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>20</td><td><b>Front Winding</b>
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.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td></td><td>FRONTWINDING_CW</td></tr><tr><td>1h</td><td></td><td>FRONTWINDING_CCW</td></tr></table></td></tr>
<tr><td></td><td>19</td><td><b>Vertex Sub Pixel Precision Select</b><table><tr><td>Format:</td><td>U1</td></tr></table>Selects the number of fractional bits maintained in the vertex data<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td></td><td>8 sub pixel precision bits maintained</td></tr><tr><td>1h</td><td></td><td>4 sub pixel precision bits maintained</td></tr></table></td></tr>
<tr><td></td><td>18</td><td><b>EarlyCull Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>This field is used to enable/disable the EarlyCull function.</td></tr>
</table>

3DSTATE_CLIP

17:16 **Cull Mode**

Format:	3D_CullMode

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

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

15:11 **Reserved**

Format:	MBZ

10 **Clipper Statistics Enable**

Format:	Enable

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

Value	Name	Description
0h	Disable	CL_INVOCATIONS_COUNT cannot increment
1h	Enable	CL_INVOCATIONS_COUNT can increment

9:8 **Reserved**

Format:	MBZ

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.

2

31 **Clip Enable**

Format:	Enable

Specifies whether the CLIP function is enabled or disabled (pass-through).

30 **API Mode**

Controls the definition of the NEAR clipping plane

Value	Name	Description
0h	APIMODE_OGL	NEAR VP boundary == 0.0 (NDC)
1h	Reserved	

29 **Reserved**

Format:	MBZ

<table>
<tr><th colspan="3">3DSTATE_CLIP</th></tr>
<tr><td></td><td>28</td><td><b>Viewport XY ClipTest Enable</b><table><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.</td></tr>
<tr><td></td><td>27</td><td><b>Viewport Z ClipTest Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>This field is used to control whether the Viewport Z extents (near, far) are considered in VertexClipTest.</td></tr>
<tr><td></td><td>26</td><td><b>Guardband ClipTest Enable</b><table><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 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.</td></tr>
<tr><td></td><td>25:24</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>23:16</td><td><b>User Clip Distance Clip Test Enable Bitmask</b><table><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 / must clip determination needs to be made. DX10 allows simultaneous use of ClipDistance and Cull Distance test of up to 8 distances.</td></tr>
<tr><td></td><td>15:13</td><td><b>Clip Mode</b>
This field specifies a general mode of the CLIP unit, when the CLIP unit is ENABLED.<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>CLIPMODE_NORMAL</td><td>TrivialAccept objects are passed down the pipeline, MustClip objects Clipped in the Fixed Function Clipper HW, TrivialReject and BAD objects are discarded</td></tr>
<tr><td>1h</td><td>Reserved</td><td></td></tr>
<tr><td>2h</td><td>Reserved</td><td></td></tr>
<tr><td>3h</td><td>CLIPMODE_REJECT_ALL</td><td>All objects are discarded</td></tr>
<tr><td>4h</td><td>CLIPMODE_ACCEPT_ALL</td><td>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.</td></tr>
<tr><td>5h-7h</td><td>Reserved</td><td></td></tr>
</table></td></tr>
<tr><td></td><td>12:10</td><td><b>Reserved</b></td></tr>
</table>

3DSTATE_CLIP

<table>
<tr><td></td><td></td><td><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>9</td><td><b>Perspective Divide Disable</b><table><tr><td>Format:</td><td>Disable</td></tr></table>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" (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).</td></tr>
<tr><td></td><td>8</td><td><b>Non-Perspective Barycentric Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>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.</td></tr>
<tr><td></td><td>7:6</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>5:4</td><td><b>Triangle Strip/List Provoking Vertex Select</b><table><tr><td>Format:</td><td>U2 enumerated type</td></tr></table>This field selects which vertex of a triangle (in a triangle strip or list primitive) is considered the "provoking vertex".<table><tr><th>Value</th><th>Name</th></tr><tr><td>0h</td><td>Vertex 0</td></tr><tr><td>1h</td><td>Vertex 1</td></tr><tr><td>2h</td><td>Vertex 2</td></tr><tr><td>3h</td><td>Reserved</td></tr></table></td></tr>
<tr><td></td><td>3:2</td><td><b>Line Strip/List Provoking Vertex Select</b><table><tr><td>Format:</td><td>U2 enumerated type</td></tr></table>This field selects which vertex of a line (in a line strip or list primitive) is considered the "provoking vertex".<table><tr><th>Value</th><th>Name</th></tr><tr><td>0h</td><td>Vertex 0</td></tr><tr><td>1h</td><td>Vertex 1</td></tr><tr><td>2h</td><td>Reserved</td></tr><tr><td>3h</td><td>Reserved</td></tr></table></td></tr>
<tr><td></td><td>1:0</td><td><b>Triangle Fan Provoking Vertex Select</b><table><tr><td>Format:</td><td>U2 enumerated type</td></tr></table>This field selects which vertex of a triangle (in a triangle fan primitive) is considered the</td></tr>
</table>

<table>
<tr><th colspan="3">3DSTATE_CLIP</th></tr>
<tr><td></td><td></td><td>"provoking vertex".<table><tr><th>Value</th><th>Name</th></tr><tr><td>0h</td><td>Vertex 0</td></tr><tr><td>1h</td><td>Vertex 1</td></tr><tr><td>2h</td><td>Vertex 2</td></tr><tr><td>3h</td><td>Reserved</td></tr></table></td></tr>
<tr><td>3</td><td>31:28</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>27:17</td><td><b>Minimum Point Width</b><table><tr><td>Format:</td><td>U8.3 pixels</td></tr></table>This value is used to clamp read-back PointWidth values.</td></tr>
<tr><td></td><td>16:6</td><td><b>Maximum Point Width</b><table><tr><td>Format:</td><td>U8.3 pixels</td></tr></table>This value is used to clamp read-back PointWidth values.</td></tr>
<tr><td></td><td>5</td><td><b>Force Zero RTAIndex Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>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.</td></tr>
<tr><td></td><td>4</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>3:0</td><td><b>Maximum VPIndex</b><table><tr><td>Format:</td><td>U4-1 index value (# of viewports)</td></tr></table>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.</td></tr>
</table>

3DSTATE_CONSTANT_DS

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
It is invalid to execute this command more than once between 3D_PRIMITIVE commands.
Constant buffers must be enabled in order from Constant Buffer 0 to Constant Buffer 3 within this command. For example, It is not allowed to enable Constant Buffer 1 by programming a non-zero value in the DS Constant Buffer 1 Read Length without a non-zero value in DS Constant Buffer 0 Read Length.

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
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2 Value | Name 5h | Excludes DWord (0,1) **[Default]**
1..6	191:0	**Constant Body** Format: 3DSTATE_CONSTANT(Body) Following table is the shared portion of the 3DSTATE_CONSTANT command for VS, HS, DS, and GS

3DSTATE_CONSTANT_GS

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
It is invalid to execute this command more than once between 3D_PRIMITIVE commands.
Constant buffers must be enabled in order from Constant Buffer 0 to Constant Buffer 3 within this command. For example, it is not allowed to enable Constant Buffer 1 by programming a non-zero value in the GS Constant Buffer 1 Read Length without a non-zero value in GS Constant Buffer 0 Read Length.

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** Format: MBZ
	14:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2 Value / Name 5h / Excludes DWord (0,1) **[Default]**
1..6	191:0	**Constant Body** Format: 3DSTATE_CONSTANT(Body) Following table is the shared portion of the 3DSTATE_CONSTANT command for VS, HS, DS, and GS

3DSTATE_CONSTANT_HS

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
It is invalid to execute this command more than once between 3D_PRIMITIVE commands.
Constant buffers must be enabled in order from Constant Buffer 0 to Constant Buffer 3 within this command. For example, It is not allowed to enable Constant Buffer 1 by programming a non-zero value in the HS Constant Buffer 1 Read Length without a non-zero value in HS Constant Buffer 0 Read Length.

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
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2 **Value** | **Name** 5h | Excludes DWord (0,1) **[Default]**
1..6	191:0	**Constant Body** Format: 3DSTATE_CONSTANT(Body) Following table is the shared portion of the 3DSTATE_CONSTANT command for VS, HS, DS, and GS

3DSTATE_CONSTANT_PS

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
It is invalid to execute this command more than once between 3D_PRIMITIVE commands.
Constant buffers must be enabled in order from Constant Buffer 0 to Constant Buffer 3 within this command. For example, it is not allowed to enable Constant Buffer 1 by programming a non-zero value in the PS Constant Buffer 1 Read Length without a non-zero value in PS Constant Buffer 0 Read Length.

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
	15:8	**Reserved** Format: MBZ
	7:0	**Dword Length** Format: =n Total Length - 2 Value: 5h — Name: Excludes DWord (0,1) **[Default]**
1..6	191:0	**Constant Body** Format: 3DSTATE_CONSTANT(Body) Following table is the shared portion of the 3DSTATE_CONSTANT command for VS, HS, DS, and GS

3DSTATE_CONSTANT_VS

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
It is invalid to execute this command more than once between 3D_PRIMITIVE commands.
Constant buffers must be enabled in order from Constant Buffer 0 to Constant Buffer 3 within this command. For example, it is not allowed to enable Constant Buffer 1 by programming a non-zero value in the VS Constant Buffer 1 Read Length without a non-zero value in VS Constant Buffer 0 Read Length.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h GFXPIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Command SubType</b><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></td><td>26:24</td><td><b>3D Command Opcode</b><table><tr><td>Default Value:</td><td>0h 3DSTATE_PIPELINED</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>23:16</td><td><b>3D Command Sub Opcode</b><table><tr><td>Default Value:</td><td>15h 3DSTATE_CONSTANT_VS</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Format:</td><td>=n Total Length - 2</td></tr></table><table><tr><th>Value</th><th>Name</th></tr><tr><td>5h</td><td>Excludes DWord (0,1) <b>[Default]</b></td></tr></table></td></tr>
<tr><td>1..6</td><td>191:0</td><td><b>Constant Body</b><table><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

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).

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).
3DSTATE_DEPTH_BUFFER must always be programmed along with the other Depth/Stencil state commands(i.e. 3DSTATE_CLEAR_PARAMS, 3DSTATE_STENCIL_BUFFER, or 3DSTATE_HIER_DEPTH_BUFFER).
The depth buffer is always Tile-Y

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b>
<table>
<tr><td>Default Value:</td><td>3h GFXPIPE</td></tr>
<tr><td>Format:</td><td>OpCode</td></tr>
</table></td></tr>
<tr><td></td><td>28:27</td><td><b>Command SubType</b>
<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></td><td>26:24</td><td><b>3D Command Opcode</b>
<table>
<tr><td>Default Value:</td><td>0h 3DSTATE_PIPELINED</td></tr>
<tr><td>Format:</td><td>OpCode</td></tr>
</table></td></tr>
<tr><td></td><td>23:16</td><td><b>3D Command Sub Opcode</b>
<table>
<tr><td>Default Value:</td><td>05h 3DSTATE_DEPTH_BUFFER</td></tr>
<tr><td>Format:</td><td>OpCode</td></tr>
</table></td></tr>
<tr><td></td><td>15:8</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>7:0</td><td><b>Dword Length</b>
<table>
<tr><td>Default Value:</td><td>5h Excludes Dword (0,1)</td></tr>
<tr><td>Format:</td><td>=n Total Length - 2</td></tr>
</table></td></tr>
<tr><td>1</td><td>31:29</td><td><b>Surface Type</b>
This field defines the type of the surface.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>SURFTYPE_1D</td><td>Defines a 1-dimensional map or array of maps</td></tr>
<tr><td>1h</td><td>SURFTYPE_2D</td><td>Defines a 2-dimensional map or array of maps</td></tr>
</table></td></tr>
</table>

3DSTATE_DEPTH_BUFFER

2h	SURFTYPE_3D	Defines a 3-dimensional (volumetric) map
3h	SURFTYPE_CUBE	Defines a cube map
4h-6h	Reserved	
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.

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

3DSTATE_DEPTH_BUFFER

<table>
<tr><td></td><td>17:0</td><td><b>Surface Pitch</b>
<table><tr><td>Format:</td><td>U18-1 Pitch in Bytes</td></tr></table>
This field specifies the pitch of the depth buffer in (#Bytes - 1).
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>[127, 3FFFFh]</td><td></td><td>corresponding to [128B, 256KB] also restricted to a multiple of 128B</td></tr></table>
<table><tr><th>Programming Notes</th></tr><tr><td>The pitch specified must be a multiple of the tile pitch, in the range [128B, 128KB].</td></tr></table>
</td></tr>
<tr><td>2</td><td>31:0</td><td><b>Surface Base Address</b>
<table><tr><td>Format:</td><td>GraphicsAddress[31:0]Depth_Buffer</td></tr></table>
This field specifies the starting Dword address of the buffer in mapped Graphics Memory.
<table><tr><th>Programming Notes</th></tr><tr><td>The Depth Buffer can only be mapped to Main Memory (uncached).</td></tr><tr><td>If the buffer is linear, the surface must be 64-byte aligned.</td></tr></table>
</td></tr>
<tr><td>3</td><td>31:18</td><td><b>Height</b>
<table><tr><td>Format:</td><td>U14</td></tr></table>
<table><tr><td>Range: SURFTYPE_1D: must be zeroSURFTYPE_2D: height of surface - 1 (y/v dimension) [0,16383]SURFTYPE_3D: height of surface - 1 (y/v dimension) [0,2047]SURFTYPE_CUBE: height of surface - 1 (y/v dimension) [0, 16383]</td></tr><tr><td>This field specifies the height of the surface. If the surface is MIP-mapped, this field contains the height of the base MIP level.</td></tr></table>
<table><tr><th>Programming Notes</th></tr><tr><td>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).</td></tr></table>
</td></tr>
<tr><td></td><td>17:4</td><td><b>Width</b>
<table><tr><td>Format:</td><td>U14-1</td></tr></table>
<table><tr><td>Range: SURFTYPE_1D: width of surface - 1 (x/u dimension) [0, 16383]SURFTYPE_2D: width of surface - 1 (x/u dimension) [0, 16383]SURFTYPE_3D: width of surface - 1 (x/u dimension) [0,2047]SURFTYPE_CUBE: width of surface - 1 (x/u dimension) [0, 16383]</td></tr><tr><td>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.</td></tr></table>
<table><tr><th>Programming Notes</th></tr><tr><td>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).</td></tr></table>
</td></tr>
</table>

3DSTATE_DEPTH_BUFFER

<table>
<tr><td></td><td>3:0</td><td><b>LOD</b>
<table><tr><td>Format:</td><td>U4 in LOD units</td></tr></table>
This field defines the MIP level that is currently being rendered into.
<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0, 14]</td><td></td></tr></table>
<table><tr><th>Programming Notes</th></tr><tr><td>The LOD of the depth buffer must be the same as the LOD of the render target(s) (defined in SURFACE_STATE)</td></tr></table>
</td></tr>
<tr><td>4</td><td>31:21</td><td><b>Depth</b>
<table><tr><td>Format:</td><td>U11-1</td></tr></table>
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.
<table><tr><th>Value</th><th>Name</th></tr>
<tr><td>[0, 2047]</td><td>SURFTYPE_1D number of array elements - 1</td></tr>
<tr><td>[0, 2047]</td><td>SURFTYPE_2D number of array elements - 1</td></tr>
<tr><td>[0, 2047]</td><td>SURFTYPE_3D depth of surface - 1 (r/z dimension)</td></tr>
<tr><td>0</td><td>SURFTYPE_CUBE (must be zero)</td></tr></table>
<table><tr><th>Programming Notes</th></tr><tr><td>The Depth of the depth buffer must be the same as the Depth of the render target(s) (defined in SURFACE_STATE).</td></tr></table>
</td></tr>
<tr><td></td><td>20:10</td><td><b>Minimum Array Element</b>
<table><tr><td>Format:</td><td>U11</td></tr></table>
<b>For 1D and 2D Surfaces:</b>
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.
<b>For 3D Surfaces:</b>
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.
<b>For Other Surfaces:</b>
This field is ignored.
<table><tr><th>Value</th><th>Name</th></tr>
<tr><td>[0, 2047]</td><td>SURFTYPE_1D/2D</td></tr>
<tr><td>[0, 2047]</td><td>SURFTYPE_3D</td></tr></table>
</td></tr>
<tr><td></td><td>9:4</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table>
</td></tr>
<tr><td></td><td>3:0</td><td><b>Depth Buffer Object Control State</b>
<table><tr><td>Format:</td><td>MEMORY_OBJECT_CONTROL_STATE</td></tr></table>
Specifies the memory object control state for the depth buffer.
</td></tr>
</table>

3DSTATE_DEPTH_BUFFER

5	31:16	**Depth Coordinate Offset Y** Format: S15 in Screen Space (pixels)(3 LSBs MBZ) Range: [-8192,8191] Bits 31:30 should be a sign extension Specifies a signed pixel offset to be added to the RenderTarget Y coordinate in order to generate a DepthBuffer Y coordinate. (See Depth Coordinate in Windower). **Programming Notes** The 3 LSBs of both offsets must be zero to ensure correct alignmentSoftware must ensure that the resulting (sum) coordinate value is non-negative This field must be zero when rendering to field-mode (interlaced) Color Buffers (i.e., when Surface State's VerticalLineStride==1). This field can only be nonzero when rendering to surfaces of type SURFTYPE_1D and SURFTYPE_2D with Depth = 0 (non-array) and LOD = 0 (non-mip mapped). The offsets need to be aligned to the hashing mode set for WM in the GT_MODE register (0x7008) bits[12:10]. For eg if the hashing mode is set to 16x16, the Depth Coordinate Y offset needs to be aligned to the 16x16 pixel block.
	15:0	**Depth Coordinate Offset X** Format: S15 in Screen Space (pixels)(3 LSBs MBZ) Range: [-8192,8191] Bits 15:14 should be a sign extension Specifies a signed pixel offset to be added to the RenderTarget X coordinate in order to generate a DepthBuffer X coordinate. (See Depth Coordinate in Windower). **Programming Notes** The 3 LSBs of both offsets must be zero to ensure correct alignmentSoftware must ensure that the resulting (sum) coordinate value is non-negative. This field must be zero when rendering to field-mode (interlaced) Color Buffers (i.e., when Surface State's VerticalLineStride==1). This field can only be nonzero when rendering to surfaces of type SURFTYPE_1D and SURFTYPE_2D with Depth = 0 (non-array) and LOD = 0 (non-mip mapped). The offsets need to be aligned to the hashing mode set for WM in the GT_MODE register (0x7008) bits[12:10]. For eg if the hashing mode is set to 16x16, the Depth Coordinate X offset needs to be aligned to the 16x16 pixel block.
6	31:21	**Render Target View Extent** Format: U11 Range: SURFTYPE_1D/2D: same value as Depth field Range: SURFTYPE_3D: [0, 2047] to indicate extent of [1, 2048] **For 3D Surfaces:** This field indicates the extent of the accessible 'R' coordinates minus 1 on the LOD currently being rendered to.

3DSTATE_DEPTH_BUFFER

		For 1D and 2D Surfaces: This field must be set to the same value as the Depth field. **For Other Surfaces:** This field is ignored.
	20:0	**Reserved** Format: MBZ

3DSTATE_DEPTH_STENCIL_STATE_POINTERS

Source:	RenderCS
Length Bias:	2

Set up the pointer to the Depth Stencil 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: 25h 3DSTATE_DEPTH_STENCIL_STATE_POINTERS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:6	**Pointer to DEPTH_STENCIL_STATE** Format: DynamicStateOffset[31:6]DEPTH_STENCIL_STATE Specifies the 64-byte aligned offset of the DEPTH_STENCIL_STATE. This offset is relative to the **Dynamic State Base Address**
	5:1	**Reserved** Format: MBZ
	0	**Reserved** Format: MB0

3DSTATE_DRAWING_RECTANGLE

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	**Reserved** Format: MBZ
	13:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 2h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:16	**Clipped Drawing Rectangle Y Min** 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** 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**

<table>
<tr><th colspan="3">3DSTATE_DRAWING_RECTANGLE</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td>[0,16383]</td><td>Device ignores bits 15:14</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>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.</td></tr>
</table>
</td></tr>
<tr><td rowspan="2">2</td><td>31:16</td><td>
<b>Clipped Drawing Rectangle Y Max</b>
<table>
<tr><td>Format:</td><td>U16 in Pixels from Color Buffer origin (upper left corner)</td></tr>
</table>
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.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>[0,16383]</td><td>Device ignores bits 31:30</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>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.</td></tr>
</table>
</td></tr>
<tr><td>15:0</td><td>
<b>Clipped Drawing Rectangle X Max</b>
<table>
<tr><td>Format:</td><td>U16 in Pixels from Color Buffer origin (upper left corner)</td></tr>
</table>
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.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>[0,16383]</td><td>Device ignores bits 15:14</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>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.</td></tr>
</table>
</td></tr>
<tr><td rowspan="2">3</td><td>31:16</td><td>
<b>Drawing Rectangle Origin Y</b>
<table>
<tr><td>Format:</td><td>S15 in Pixels from Color Buffer origin (upper left corner).</td></tr>
</table>
<table>
<tr><th>Description</th></tr>
<tr><td>Range: [-16384,16383] (Bit 31 should be a sign extension)</td></tr>
<tr><td>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.</td></tr>
</table>
</td></tr>
<tr><td>15:0</td><td>
<b>Drawing Rectangle Origin X</b>
<table>
<tr><td>Format:</td><td>S15 in Pixels from Color Buffer origin (upper left corner).</td></tr>
</table>
<table>
<tr><th>Description</th></tr>
<tr><td>Range: [-16384,16383] (Bit 15 should be a sign extension)</td></tr>
</table>
</td></tr>
</table>

3DSTATE_DRAWING_RECTANGLE		
		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

<table>
<tr><td>Source:</td><td colspan="2">RenderCS</td></tr>
<tr><td>Length Bias:</td><td colspan="2">2</td></tr>
<tr><td colspan="3">The state used by DS is defined with this inline state packet</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h GFXPIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Command SubType</b><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></td><td>26:24</td><td><b>3D Command Opcode</b><table><tr><td>Default Value:</td><td>0h 3DSTATE_PIPELINED</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>23:16</td><td><b>3D Command Sub Opcode</b><table><tr><td>Default Value:</td><td>1Dh 3DSTATE_DS</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>4h Excludes DWord (0,1)</td></tr><tr><td>Format:</td><td>=n Total Length - 2</td></tr></table></td></tr>
<tr><td>1</td><td>31:6</td><td><b>Kernel Start Pointer</b><table><tr><td>Format:</td><td>InstructionBaseOffset[31:6]Kernel</td></tr></table>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.</td></tr>
<tr><td></td><td>5:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>2</td><td>31</td><td><b>Single Domain Point Dispatch</b><table><tr><td>Format:</td><td>U1 Enumerated Type</td></tr></table>This field can be used to force single domain point SIMD4x2 DS threads.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Multiple</td><td>Dual domain point SIMD4x2 thread dispatches are allowed.</td></tr><tr><td>1h</td><td>Single</td><td>Single domain point SIMD4x2 thread dispatches are forced.</td></tr></table></td></tr>
<tr><td></td><td>30</td><td><b>Vector Mask Enable</b><table><tr><td>Format:</td><td>U1 Enumerated Type</td></tr></table>When SPF=0, Vector Mask Enable (VME) specifies which mask to use to initialize the initial</td></tr>
</table>

3DSTATE_DS

channel enables. When SPF=1, VME specifies which mask to use to generate execution channel enables.

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

29:27 **Sampler Count**

Format:	U3

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

This field is ignored if DS Function Enable is DISABLED.

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

26 **Reserved**

Format:	MBZ

25:18 **Binding Table Entry Count**

Format:	U8

When HW Generated Binding Table is disabled:

Specifies how many binding table entries the kernel uses. Used only for prefetching of the binding table entries and associated surface state.

Note: For kernels using a large number of binding table entries, it may be wise to set this field to zero to avoid prefetching too many entries and thrashing the state cache.

This field is ignored if DS Function Enable is DISABLED.

Value	Name
[0,255]	

17 **Reserved**

Format:	MBZ

16 **Floating Point Mode**

Format:	U1 Enumerated Type

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

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

15 **Reserved**

Format:	MBZ

14 **Reserved**

3DSTATE_DS

<table>
<tr><td></td><td></td><td><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>13</td><td><b>Illegal Opcode Exception Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>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.</td></tr>
<tr><td></td><td>12:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7</td><td><b>Software Exception Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>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.</td></tr>
<tr><td></td><td>6:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>3</td><td>31:10</td><td><b>Scratch Space Base Offset</b><table><tr><td>Format:</td><td>GeneralStateOffset[31:10]ScratchSpace</td></tr></table>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.</td></tr>
<tr><td></td><td>9:4</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>3:0</td><td><b>Per-Thread Scratch Space</b><table><tr><td>Format:</td><td>U4 power of 2 Bytes over 1K Bytes</td></tr></table>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.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,11]</td><td>indicating [1K Bytes, 2M Bytes]</td></tr></table><table><tr><th>Programming Notes</th></tr><tr><td>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.</td></tr></table></td></tr>
</table>

<table>
<tr><th colspan="3">3DSTATE_DS</th></tr>
<tr><td>4</td><td>31:25</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>24:20</td><td><b>Dispatch GRF Start Register For URB Data</b><table><tr><td>Format:</td><td>U5</td></tr></table>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.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,31]</td><td>indicating GRF [R0,R31]</td></tr></table></td></tr>
<tr><td></td><td>19:18</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>17:11</td><td><b>Patch URB Entry Read Length</b><table><tr><td>Format:</td><td>U7</td></tr></table>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.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0, 64]</td><td></td></tr></table></td></tr>
<tr><td></td><td>10</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>9:4</td><td><b>Patch URB Entry Read Offset</b>
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.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0, 63]</td><td></td></tr></table></td></tr>
<tr><td></td><td>3:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>5</td><td>31:25</td><td><b>Maximum Number of Threads</b><table><tr><td>Format:</td><td>U7-1 Thread Count</td></tr></table>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.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>[0,15]</td><td></td><td>indicating thread count of [1,16]</td></tr></table></td></tr>
<tr><td></td><td>24:21</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>20:11</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>10</td><td><b>Statistics Enable</b></td></tr>
</table>

3DSTATE_DS

		Format: Enable If ENABLED, this FF unit will engage in statistics gathering. If DISABLED, statistics information associated with this FF stage will be left unchanged. This field is ignored if DS Function Enable is DISABLED.
	9:3	**Reserved**
	2	**Compute W Coordinate Enable** Format: Enable If ENABLED, the DS unit will (for each domain point) compute W = 1 - (U + V) and pass the result as a floating point value in the DS thread payload. If DISABLED, 0.0 will be passed. This field must only be ENABLED for the tessellation of TRI domains, where UVW coordinates are required. This field must be DISABLED for other domains (as they only require UV coordinates) otherwise the computed W coordinate is UNDEFINED. This field is ignored if DS Function Enable is DISABLED.
	1	**DS Cache Disable** Format: Disable This bit controls the operation of the DS Cache. This field is ignored if DS Function Enable is DISABLED. If the DS Cache is DISABLED and the DS Function is ENABLED, the DS Cache is not used and all incoming domain points will be passed to DS threads. If the DS Cache is ENABLED and the DS Function is ENABLED, incoming domain points that do not hit in the DS Cache will be passed to DS threads. The DS Cache is invalidated whenever the DS Cache becomes DISABLED , whenever the DS Function Enable toggles, and between patches.
	0	**DS Function Enable** Format: Enable If ENABLED, DS threads will be spawned to process incoming domain points which miss in the DS cache. If DISABLED, the DS stage goes into pass-through mode and performs no specific processing. This field is always used. **Programming Notes** The tessellation stages (HS, TE and DS) must be enabled/disabled as a group. I.e., draw commands can only be issued if all three stages are enabled or all three stages are disabled, otherwise the behavior is UNDEFINED.

3DSTATE_DS

Source:	RenderCS
Length Bias:	2

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 1Dh 3DSTATE_DS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 4h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:6	**Kernel Start Pointer** Format: InstructionBaseOffset[31:6]Kernel This field specifies the starting location of the kernel program run by threads spawned by this FF unit. It is specified as a 64-byte-granular offset from the Instruction Base Address. This field is ignored if DS Function Enable is DISABLED.
	5:0	**Reserved** Format: MBZ
2	31	**Single Program Flow (SPF)** Format: U1 Enumerated Type 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	Multiple	Multiple Program Flows (1- or 2-vertex threads spawned, operating under normal (SIMD4x2) mode)
1h	Single	Single Program Flow (only 1-vertex threads spawned, operating under SPF EU mode)

3DSTATE_DS

30 **Vector Mask Enable (VME)**

Format:	U1 Enumerated Type

When SPF=0, 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
0h	Dmask	Channels are enabled based on the dispatch mask
1h	Vmask	Channels are enabled based on the vector mask

29:27 **Sampler Count**

Format:	U3

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

This field is ignored if DS Function Enable is DISABLED.

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

26 **Reserved**

Format:	MBZ

25:18 **Binding Table Entry Count**

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.

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

Programming Notes
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.

17 **Thread Priority**

Format:	U1 Enumerated Type

Specifies the priority of the thread for dispatch:

This field is ignored if DS Function Enable is DISABLED.

Value	Name
0h	Normal Priority
1h	High Priority

16 **Floating Point Mode**

Format:	U1 Enumerated Type

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

This field is ignored if DS Function Enable is DISABLED.

3DSTATE_DS

		Value | **Name** | **Description** 0h | IEEE-754 | Use IEEE-754 Rules 1h | Alternate | Use alternate rules
	15:14	**Reserved** Format: | MBZ
	13	**Illegal Opcode Exception Enable** Format: | Enable This bit gets loaded into EU CR0.1[12] (note the bit # difference). See Exceptions and ISA Execution Environment. This field is ignored if DS Function Enable is DISABLED.
	12:8	**Reserved** Format: | MBZ
	7	**Software Exception Enable** Format: | Enable This bit gets loaded into EU CR0.1[13] (note the bit # difference). See Exceptions and ISA Execution Environment. This field is ignored if DS Function Enable is DISABLED.
	6:0	**Reserved** Format: | MBZ
3	31:10	**Scratch Space Base Offset** Format: | GeneralStateOffset[31:10]ScratchSpace Specifies the starting location of the scratch space area allocated to this FF unit as a 1K-byte aligned offset from the General State Base Address. If required, each thread spawned by this FF unit will be allocated some portion of this space, as specified by Per-Thread Scratch Space. The computed offset of the thread-specific portion will be passed in the thread payload as Scratch Space Offset. The thread is expected to utilize stateless DataPort read/write requests to access scratch space, where the DataPort will cause the General State Base Address to be added to the offset passed in the request header. This field is ignored if DS Function Enable is DISABLED.
	9:4	**Reserved** Format: | MBZ
	3:0	**Per-Thread Scratch Space** Format: | U4 power of 2 Bytes over 1K Bytes Specifies the amount of scratch space to be allocated to each thread spawned by this FF unit. The driver must allocate enough contiguous scratch space, starting at the Scratch Space Base Pointer, to ensure that the Maximum Number of Threads can each get Per-Thread Scratch Space size without exceeding the driver-allocated scratch space. This field is ignored if DS Function Enable is DISABLED. **Value** | **Name**

3DSTATE_DS

<table>
<tr><td></td><td></td><td><table><tr><td>[0,11]</td><td>indicating [1K Bytes, 2M Bytes]</td></tr></table><table><tr><th>Programming Notes</th></tr><tr><td>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.</td></tr></table></td></tr>
<tr><td>4</td><td>31:25</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>24:20</td><td><b>Dispatch GRF Start Register for URB Data</b><table><tr><td>Format:</td><td>U5</td></tr></table>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.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,31]</td><td>indicating GRF [R0,R31]</td></tr></table></td></tr>
<tr><td></td><td>19:18</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>17:11</td><td><b>Patch URB Entry Read Length</b><table><tr><td>Format:</td><td>U7</td></tr></table>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.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,64]</td><td></td></tr></table></td></tr>
<tr><td></td><td>10</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>9:4</td><td><b>Patch URB Entry Read Offset</b>
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.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0, 63]</td><td></td></tr></table></td></tr>
<tr><td></td><td>3:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>5</td><td>31:25</td><td><b>Maximum Number of Threads</b><table><tr><td>Format:</td><td>U7-1 representing thread count</td></tr></table>Specifies the maximum number of simultaneous DS threads allowed to be active. Used to avoid using up the scratch space, or to avoid potential deadlock.
This field is ignored if DS Function Enable is DISABLED.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,15]</td><td>indicating thread count of [1,16]</td></tr></table></td></tr>
</table>

3DSTATE_DS

	24:11	**Reserved** Format: MBZ
	10	**Statistics Enable** Format: Enable If ENABLED, this FF unit will engage in statistics gathering. If DISABLED, statistics information associated with this FF stage will be left unchanged. This field is ignored if DS Function Enable is DISABLED.
	9:3	**Reserved** Format: MBZ
	2	**Compute W Coordinate Enable** Format: Enable If ENABLED, the DS unit will (for each domain point) compute W = 1 (U + V) and pass the result as a floating point value in the DS thread payload. If DISABLED, 0.0 wil 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	**DS Cache Disable** Format: Disable This bit controls the operation of the DS Cache. This field is ignored if DS Function Enable is DISABLED. If the DS Cache is DISABLED and the DS Function is ENABLED, the DS Cache is not used and all incoming domain points will be passed to DS threads. If the DS Cache is ENABLED and the DS Function is ENABLED, incoming domain points that do not hit in the DS Cache will be passed to DS threads. The DS Cache is invalidated whenever the DS Cache becomes DISABLED , whenever the DS Function Enable toggles, and between patches.
	0	**DS Function Enable** Format: Enable If ENABLED, DS threads will be spawned to process incoming domain points which miss in the DS cache. If DISABLED, the DS stage goes into pass-through mode and performs no specific processing. This field is always used. **Programming Notes** The tessellation stages (HS, TE and DS) must be enabled/disabled as a group. I.e., draw commands can only be issued if all three stages are enabled or all three stages are disabled, otherwise the behavior is UNDEFINED.

3DSTATE_GATHER_CONSTANT_HS

<table>
<tr><td>Source:</td><td>RenderCS</td></tr>
<tr><td>Length Bias:</td><td>2</td></tr>
</table>

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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h GFXPIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Command SubType</b><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></td><td>26:24</td><td><b>3D Command Opcode</b><table><tr><td>Default Value:</td><td>0h 3DSTATE_PIPELINED</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>23:16</td><td><b>3D Command Sub Opcode</b><table><tr><td>Default Value:</td><td>36h 3DSTATE_GAHTER_CONSTANT_HS</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>1h - 80h Excludes DWord (0,1)</td></tr><tr><td>Format:</td><td>=n</td></tr></table>Total Length - 2</td></tr>
<tr><td>1</td><td>31:16</td><td><b>Constant Buffer Valid</b><table><tr><td>Format:</td><td>U16</td></tr></table>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.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,65535]</td><td></td></tr></table></td></tr>
<tr><td></td><td>15:12</td><td><b>Constant Buffer Binding Table Block</b><table><tr><td>Format:</td><td>U4</td></tr></table>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.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,15]</td><td></td></tr></table></td></tr>
<tr><td></td><td>11:2</td><td><b>Reserved</b></td></tr>
</table>

3DSTATE_GATHER_CONSTANT_HS

		Format: MBZ
	1	**Reserved** 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. **Programming Notes** SW increments the offset by the size of the gather buffer in 512 bit units for each gather buffer generated.
	5	**Reserved** Format: MBZ
	4	**Reserved** Format: MBZ
	3	**Reserved** Format: MBZ
	2:0	**Reserved** Format: MBZ
3..n	31:24	**Constant Buffer Offset for Entry 2n+1** Format: Offset[7:0] This field specifies the Offset in 128-bit units of the 128b entry fetched from the constant buffer for entry 1. Surface Type:ConstantBuffer
	23:20	**Channel Mask for Entry 2n+1** Mask: Mask[3:0] Format: ConstantBuffer Each bit of this field correspond to the 4 channels of each entry fetched from memory. When the bit is a 1, the corresponding 32-bit value is loaded in FF's push constant buffer. When the bit is a 0, the corresponding 32-bit value is not loaded. If this field is zero it means the entry is not used. **Programming Notes** This field may only be zero if it is the last dword of the command packet.
	19:16	**Binding Table Index Offset for Entry 2n+1** Format: Constant Buffer Index offset [3:0]Surface State for ConstantBuffer This field specifies the Binding Table index offset from the **Constant Buffer Binding Table Block** starting point in the Binding Table. This value is added to the **Constant Buffer Binding Table Block** will result in the Binding Table Index pointing to the surface state containing the constant buffer to be referenced.

3DSTATE_GATHER_CONSTANT_HS

	15:8	**Constant Buffer Offset for Entry 2n+0** Format: Offset[7:0]ConstantBuffer This field specifies the Offset in 128-bit units of the 128b entry fetched from the constant buffer for entry 2n+0 (including when **On-Die Table Read Enable** is set).
	7:4	**Channel Mask for Entry 2n+0** Mask: Mask[3:0] Format: ConstantBuffer Each bit of this field correspond to the 4 channels of each entry fetched from memory. When the bit is a 1, the corresponding 32-bit value is loaded in FF's push constant buffer. When the bit is a 0, the corresponding 32-bit value is not loaded. If this field is zero it means the entry is not used.
	3:0	**Binding Table Index offset for Entry 2n+0** Format: Constant Buffer Index offset [3:0]Surface State for ConstantBuffer This field specifies the Binding Table index offset from the **Constant Buffer Binding Table Block** starting point in the Binding Table. This value is added to the **Constant Buffer Binding Table Block** will result in the Binding Table Index pointing to the surface state containing the constant buffer to be referenced.

3DSTATE_GS

Source:	RenderCS
Length Bias:	2

Controls the GS stage hardware.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 11h 3DSTATE_GS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 5h Excludes DWord (0,1) Format: =n
1	31:6	**Kernel Start Pointer** Format: InstructionBaseOffset[31:6]Kernel This field specifies the starting location (1st GEN4 core instruction) of the kernel program run by threads spawned by this FF unit. It is specified as a 64-byte-granular offset from the Instruction Base Address.
	5:0	**Reserved** Format: MBZ
2	31	**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** 0h | Disable | Single Program Flow disabled 1h | Enable | Single Program Flow enabled
	30	**Vector Mask Enable (VME)** Format: U1 enumerated type When SPF=0, VME specifies which mask to use to initialize the initial channel enables. When

3DSTATE_GS

SPF=1, VME specifies which mask to use to generate execution channel enables.

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

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	Reserved

26 **Reserved**

Format:	MBZ

25:18 **Binding Table Entry Count**

Format:	U8

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

17 **Thread Priority**

Specifies the priority of the thread for dispatch

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

16 **Floating Point Mode**

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

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

15:14 **Reserved**

Format:	MBZ

13 **Illegal Opcode Exception Enable**

Format:	Enable
Double Buffer Armed By:	This bit gets loaded into EU CR0.1[12] (note the bit # difference). See Exceptions and ISA Execution Environment.

3DSTATE_GS

	12	**Reserved** Format: MBZ
	11	**Mask Stack Exception Enable** Format: Enable Double Buffer Armed By: 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.
	6	**Reserved** Format: MBZ
	5:0	**Reserved** Format: MBZ
3	31:10	**Scratch Space Base Pointer** Format: GeneralStateOffset[31:10] Specifies the location of the scratch space area allocated to this FF unit, specified as a 1KB-granular offset from the General State Base Address. If required, each thread spawned by this FF unit will be allocated some portion of this space, as specified by Per-Thread Scratch Space.
	9:4	**Reserved** Format: MBZ
	3:0	**Per-Thread Scratch Space** Format: U4 Power of 2 Bytes over 1K Bytes Specifies the amount of scratch space to be allocated to each thread spawned by this FF unit.The driver must allocate enough contiguous scratch space, starting at the Scratch Space Base Pointer, to ensure that the Maximum Number of Threads can each get Per-Thread Scratch Space size without exceeding the driver-allocated scratch space. **Value** — **Name** [0,11] — indicating [1K Bytes, 2M Bytes]
4	31:29	**Reserved** Format: MBZ
	28:23	**Output Vertex Size** Format: U6 [0,62] indicating [1,63] 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).

3DSTATE_GS

<table>
<tr><td></td><td></td><td>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>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.</td></tr>
</table>
</td></tr>
<tr><td></td><td>22:17</td><td>
<b>Output Topology</b>
<table>
<tr><td>Format:</td><td>3DPrimType</td></tr>
</table>
This field specifies the topology type (3DPrimType) to be associated with GS-thread output vertices (if any).
</td></tr>
<tr><td></td><td>16:11</td><td>
<b>Vertex URB Entry Read Length</b>
Specifies the amount of URB data read and passed in the thread payload for each Vertex URB entry, in 256-bit register increments.
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>Programming Restriction:This field must be a non-zero value if Include Vertex Handles is cleared to zero.</td></tr>
</table>
</td></tr>
<tr><td></td><td>10</td><td>
<b>Include Vertex Handles</b>
<table>
<tr><td>Format:</td><td>Boolean</td></tr>
</table>
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.
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>This field must be set if Vertex URB Entry Read Length is cleared to zero.</td></tr>
<tr><td>When this field is set and GS is enabled, only PATCHLIST topologies may be submitted. I.e., pull-model vertices are only supported for PATCH objects, other object types must completely push all vertex data into the payload.</td></tr>
</table>
</td></tr>
<tr><td></td><td>9:4</td><td>
<b>Vertex URB Entry Read Offset</b>
<table>
<tr><td>Double Buffer Armed By:</td><td>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.</td></tr>
</table>
</td></tr>
<tr><td></td><td>3:0</td><td>
<b>Dispatch GRF Start Register for URB Data</b>
<table>
<tr><td>Format:</td><td>U4</td></tr>
</table>
Specifies the starting GRF register number for the URB portion (Constant + Vertices) of the thread payload.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>[0,15]</td><td>indicating GRF [R0,R15]</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>If Include Vertex Handles is enabled (pull or hybrid handles case), then
For DUAL_OBJECT dispatch mode this field should be:</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th colspan="3">3DSTATE_GS</th></tr>
<tr><td></td><td></td><td>(((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</td></tr>
<tr><td>5</td><td>31:25</td><td><b>Maximum Number of Threads</b>
<table><tr><td>Format:</td><td>U7-1 thread count</td></tr></table>
Specifies the maximum number of simultaneous threads allowed to be active. Used to avoid using up the scratch space, or to avoid potential deadlock.
<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,15]</td><td>indicating thread count of [1,16]</td></tr></table></td></tr>
<tr><td></td><td>24</td><td><b>Control Data Format</b>
<table><tr><td>Format:</td><td>U1</td></tr></table>
This field specifies the format of the control data header (if any).
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>GSCTL_CUT</td><td>The control data header contains cut bits.</td></tr><tr><td>1h</td><td>GSCTL_SID</td><td>The control data header contains StreamID bits. . Output Topology must be set to POINTLIST, or behavior is UNDEFINED.</td></tr></table></td></tr>
<tr><td></td><td>23:20</td><td><b>Control Data Header Size</b>
<table><tr><td>Format:</td><td>U4</td></tr></table>
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.
<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,8]</td><td>32B units</td></tr></table></td></tr>
<tr><td></td><td>19:15</td><td><b>Instance Control</b>
<table><tr><td>Format:</td><td>U5-1 in #instances</td></tr></table>
<table><tr><td>[0,31] indicating [1,32] instances</td></tr><tr><td>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.</td></tr></table></td></tr>
<tr><td></td><td>14:13</td><td><b>Default StreamID</b>
<table><tr><td>Format:</td><td>U2</td></tr></table>
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</td></tr>
</table>

<table>
<tr><th colspan="3">3DSTATE_GS</th></tr>
<tr><td></td><td></td><td>GS is disabled, StreamID will be output as 0.</td></tr>
<tr><td></td><td>12:11</td><td><b>Dispatch Mode</b>
<table><tr><td>Format:</td><td>U2</td></tr></table>
This field specifies how the GS unit dispatches multiple instances and/or multiple objects.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>SINGLE</td><td>Each thread shades a single instance of one object.</td></tr>
<tr><td>1h</td><td>DUAL_INSTANCE</td><td>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. The GS must be allocated at least two URB handles or behavior is UNDEFINED.</td></tr>
<tr><td>2h</td><td>DUAL_OBJECT</td><td>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). The GS must be allocated at least two URB handles or behavior is UNDEFINED.</td></tr>
<tr><td>3h</td><td>Reserved</td><td></td></tr>
</table></td></tr>
<tr><td></td><td>10</td><td><b>GS Statistics Enable</b>
This bit controls whether GS-unit-specific statistics register(s) can be incremented.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Disable</td><td>GS_INVOCATIONS_COUNT and GS_PRIMITIVES_COUNT cannot increment</td></tr>
<tr><td>1h</td><td>Enable</td><td>GS_INVOCATIONS_COUNT and GS_PRIMITIVES_COUNT can increment</td></tr>
</table></td></tr>
<tr><td></td><td>9:5</td><td><b>GS Invocations Increment Value</b>
<table><tr><td>Format:</td><td>U5</td></tr></table>
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.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>[0,31]</td><td>indicating an increment of [1,32]</td></tr>
</table></td></tr>
<tr><td></td><td>4</td><td><b>Include Primitive ID</b>
<table><tr><td>Format:</td><td>Boolean</td></tr></table>
If set, R1 of the payload is written with Primitive ID value(s). If clear, these Primitive ID values are</td></tr>
</table>

3DSTATE_GS

		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.
	2	**Reorder Enable** Format: Enable This bit controls whether the GS unit reorders TRISTRIP/TRISTRIP_REV vertices passed in the GS thread payload. If ENABLED, the GS unit will reorder the vertices for "odd-numbered" triangles originating from TRISTRIP topologies and "even-numbered" triangles originating from TRISTRIP_REV topologies. (Note that the first triangle is considered "triangle 0", which is even-numbered). With respect to the PrimType passed in the GS thread payload, the GS unit passes TRISTRIP when the vertices are not reordered, and TRISTRIP_REV when the vertices are reordered (regardless of whether a TRISTRIP or TRISTRIP_REV topology was being processed)If DISABLED, TRISTRIP/TRISTRIP_REV vertices are not reordered, and always passed in the order they are received from the pipeline. The GS unit will still toggle PrimType on alternating (as described above) so that the GS thread can perform the reordering internally (or do whatever is necessary to account for the non-reordering of its input).
	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

3DSTATE_GS

		should be submitted, otherwise the GS kernel will produce unpredictable results.) Also note that it is possible to craft a GS kernel which can accept any object type that's thrown at it by first examining the PrimType passed in the payload and then using this info to correctly interpret the number of vertices passed in the payload.
	0	**GS Enable** Format: Enable Specifies whether the GS stage is enabled or disabled (pass-through).
6	31	**Reserved** Format: MBZ
	30:13	**Reserved** Format: MBZ
	12	**Reserved** Format: MBZ
	11:0	**Semaphore Handle** Format: URBOffset[17:6] This is the URB offset pointing to the first of the GS semaphore DWords in the URB. The size of the region is 128 DWs(8 - 512b URB entries). Software is responsible for allocating combined GS and/or HS semaphore Dwords in a single contiguous region of the URB. Software must also make sure the 3D pipeline is IDLE prior to allocating or deallocating the region. The semaphores can be located in an unused area within a FF unit's URB fenced region or an unused area within the Push Constant region.

3DSTATE_GS

Source:	RenderCS
Length Bias:	2

Controls the GS stage hardware.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h GFXPIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Command SubType</b><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></td><td>26:24</td><td><b>3D Command Opcode</b><table><tr><td>Default Value:</td><td>0h 3DSTATE_PIPELINED</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>23:16</td><td><b>3D Command Sub Opcode</b><table><tr><td>Default Value:</td><td>11h 3DSTATE_GS</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>5h Excludes DWord (0,1)</td></tr><tr><td>Format:</td><td>=n</td></tr></table></td></tr>
<tr><td>1</td><td>31:6</td><td><b>Kernel Start Pointer</b><table><tr><td>Format:</td><td>InstructionBaseOffset[31:6]Kernel</td></tr></table>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 <b>Instruction Base Address</b>.</td></tr>
<tr><td></td><td>5:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>2</td><td>31</td><td><b>Single Program Flow</b><table><tr><td>Format:</td><td>Enable</td></tr></table>Specifies the initial condition of the kernel program as either a single program flow (SPF) (SIMDnxm with m = 1) or as multiple program flows (SIMDnxm with m > 1). See CR0 description in <i>ISA Execution Environment</i>.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td>Disable</td><td>Single Program Flow disabled</td></tr><tr><td>1</td><td>Enable</td><td>Single Program Flow enable</td></tr></table></td></tr>
<tr><td></td><td>30</td><td><b>Vector Mask Enable</b>
When SPF=0, Vector Mask Enable (VME) specifies which mask to use to initialize the initial</td></tr>
</table>

3DSTATE_GS

channel enables. When SPF=1, VME specifies which mask to use to generate execution channel enables.

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

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**

Format:	MBZ

25:18 **Binding Table Entry Count**

Format:	U8

When **HW Generated Binding Table** is disabled:
Specifies how many binding table entries the kernel uses. Used only for prefetching of the binding table entries and associated surface state.

Programming Notes
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.

17 **Thread Priority**

Specifies the priority of the thread for dispatch

Value	Name
0h	Normal Priority
1h	High Priority

16 **Floating Point Mode**

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

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

15:14 **Reserved**

3DSTATE_GS

<table>
<tr><td></td><td></td><td><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>13</td><td><b>Illegal Opcode Exception Enable</b><table><tr><td>Format:</td><td>Enable</td></tr><tr><td>Double Buffer Armed By:</td><td>This bit gets loaded into EU CR0.1[12] (note the bit # difference). See Exceptions and ISA Execution Environment.</td></tr></table></td></tr>
<tr><td></td><td>12</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11</td><td><b>Mask Stack Exception Enable</b><table><tr><td>Format:</td><td>Enable</td></tr><tr><td>Double Buffer Armed By:</td><td>This bit gets loaded into EU CR0.1[11]. See Exceptions and ISA Execution Environment.</td></tr></table>This bit gets loaded into EU CR0.1[11]. See <i>Exceptions</i> and <i>ISA Execution Environment</i>.</td></tr>
<tr><td></td><td>10:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7</td><td><b>Software Exception Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>This bit gets loaded into EU CR0.1[13] (note the bit # difference). See <i>Exceptions</i> and <i>ISA Execution Environment</i>.</td></tr>
<tr><td></td><td>6:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>3</td><td>31:10</td><td><b>Scratch Space Base Pointer</b><table><tr><td>Format:</td><td>GeneralStateOffset[31:10]</td></tr></table>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.</td></tr>
<tr><td></td><td>9:4</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>3:0</td><td><b>Per-Thread Scratch Space</b><table><tr><td>Format:</td><td>U4 power of 2 Bytes over 1K Bytes</td></tr></table>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.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>[0,11]</td><td></td><td>indicating [1K Bytes, 2M Bytes]</td></tr></table></td></tr>
<tr><td>4</td><td>31:29</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>28:23</td><td><b>Output Vertex Size</b></td></tr>
</table>

3DSTATE_GS

<table>
<tr><td></td><td></td><td>
<table><tr><td>Format:</td><td>U6</td></tr></table>
<table><tr><td>[0,63] indicating [1,64] 16B units</td></tr><tr><td>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).</td></tr></table>
<table><tr><th>Programming Notes</th></tr><tr><td>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.</td></tr></table>
</td></tr>
<tr><td></td><td>22:17</td><td><b>Output Topology</b>
<table><tr><td>Format:</td><td>3DPrimType</td></tr></table>
This field specifies the topology type (3DPrimType) to be associated with GS-thread output vertices (if any).</td></tr>
<tr><td></td><td>16:11</td><td><b>Vertex URB Entry Read Length</b>
Specifies the amount of URB data read and passed in the thread payload for each Vertex URB entry, in 256-bit register increments.
<table><tr><th>Programming Notes</th></tr><tr><td>This field must be a non-zero value if <b>Include Vertex Handles</b> is cleared to zero.</td></tr></table>
</td></tr>
<tr><td></td><td>10</td><td><b>Include Vertex Handles</b>
<table><tr><td>Format:</td><td>Boolean</td></tr></table>
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.
<table><tr><th>Programming Notes</th></tr><tr><td>When set (and the GS is enabled) , only PATCHLIST topologies can be issued to the pipeline, or operation is UNDEFINED.
This field must be set if <b>Vertex URB Entry Read Length</b> is cleared to zero.</td></tr></table>
</td></tr>
<tr><td></td><td>9:4</td><td><b>Vertex URB Entry Read Offset</b>
<table><tr><td>Double Buffer Armed By:</td><td>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.</td></tr></table>
</td></tr>
<tr><td></td><td>3:0</td><td><b>Dispatch GRF Start Register for URB Data</b>
<table><tr><td>Format:</td><td>U4</td></tr></table>
Specifies the starting GRF register number for the URB portion (Constant + Vertices) of the thread payload.
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>[0,15]</td><td></td><td>indicating GRF [R0,R15]</td></tr></table>
</td></tr>
</table>

3DSTATE_GS

Programming Notes
If Include Vertex Handles is enabled (pull or hybrid handles case), then 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

5 31:25 **Maximum Number of Threads**

Format:	U7
Format:	U7-1 thread count

Specifies the maximum number of simultaneous threads allowed to be active. Used to avoid using up the scratch space, or to avoid potential deadlock.

Value	Name
[0,15]	indicating thread count of [1,16]
[0,35]	indicating thread count of [1,36]

24 **Control Data Format**

Format:	U1

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

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

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.

Value	Name
[0,8]	32B units

19:15 **Instance Count**

Format:	U5-1 in #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 StreamID**

3DSTATE_GS

Format:	U2

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

12:11 **Dispatch Mode**

Format:	U2

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

Value	Name	Description
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. The GS must be allocated at least two URB handles or behavior is UNDEFINED.
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). The GS must be allocated at least two URB handles or behavior is UNDEFINED.
3h	Reserved	

10 **GS Statistics 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 **GSInvocations 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

3DSTATE_GS

		[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.
	2	**Reorder Enable** Format: | Enable This bit controls whether the GS unit reorders TRISTRIP/TRISTRIP_REV vertices passed in the GS thread payload. If ENABLED, the GS unit will reorder the vertices for odd-numbered triangles originating from TRISTRIP topologies and even-numbered triangles originating from TRISTRIP_REV topologies. (Note that the first triangle is considered triangle 0, which is even-numbered). With respect to the PrimType passed in the GS thread payload, the GS unit passes TRISTRIP when the vertices are not reordered, and TRISTRIP_REV when the vertices are reordered (regardless of whether a TRISTRIP or TRISTRIP_REV topology was being processed) If DISABLED, TRISTRIP/TRISTRIP_REV vertices are not reordered, and always passed in the order they are received from the pipeline. The GS unit will still toggle PrimType on alternating (as described above) so that the GS thread can perform the reordering internally (or do whatever is necessary to account for the non-reordering of its input).
	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,

3DSTATE_GS

		software should set this bit just in case a TRILIST_ADJ is submitted to the pipeline.) Note that the GS unit is otherwise not aware of the object type that is expected by the GS kernel. It is up to software to ensure that the submitted primitive type (in 3DPRIMITIVE) is otherwise compatible with the object type expected by the GS kernel. (E.g., if the GS kernel expects a LINE_ADJ object, only LINELIST_ADJ or LINESTRIP_ADJ should be submitted, otherwise the GS kernel will produce unpredictable results.) Also note that it is possible to craft a GS kernel which can accept any object type that's thrown at it by first examining the PrimType passed in the payload and then using this info to correctly interpret the number of vertices passed in the payload.
	0	**GS Enable** Format: Enable Specifies whether the GS stage is enabled or disabled (pass-through).
6	31:12	**Reserved** Format: MBZ
	11:0	**Semaphore Handle** Format: U12 Handle This is the 512b-aligned URB handle pointing to the first of the GS semaphore DWords in the URB. Software is responsible for statically allocating combined GS and/or HS semaphore Dwords in a single contiguous region of the URB. The semaphores can be located in an unused area within a FF unit's URB fenced region or an unused area within the Push Constant region.

3DSTATE_HIER_DEPTH_BUFFER

Source:	RenderCS
Length Bias:	2

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).

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).
3DSTATE_HIER_DEPTH_BUFFER must always be programmed in the along with the other Depth/Stencil state commands(i.e. 3DSTATE_DEPTH_BUFFER, 3DSTATE_CLEAR_PARAMS, or 3DSTATE_STENCIL_BUFFER)

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h GFXPIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Command SubType</b><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></td><td>26:24</td><td><b>3D Command Opcode</b><table><tr><td>Default Value:</td><td>0h 3DSTATE_PIPELINED</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>23:16</td><td><b>3D Command Sub Opcode</b><table><tr><td>Default Value:</td><td>07h 3DSTATE_HIER_DEPTH_BUFFER</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>Dword Length</b><table><tr><td>Format:</td><td>=n Total Length - 2</td></tr></table><table><tr><th>Value</th><th>Name</th></tr><tr><td>1h</td><td>Excludes Dword (0,1) <b>[Default]</b></td></tr></table></td></tr>
<tr><td>1</td><td>31:29</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>28:25</td><td><b>Hierarchical Depth Buffer Object Control State</b><table><tr><td>Format:</td><td>MEMORY_OBJECT_CONTROL_STATE</td></tr></table></td></tr>
</table>

3DSTATE_HIER_DEPTH_BUFFER

Description
Specifies the memory object control state for the hierarchical depth buffer.
This field is not context save and restored by hardware. If this field is programmed to any value other than zero, it must be programmed after the following commands or events: • MI_SET_CONTEXT • MI_WAIT_FOR_EVENT (Specifically waits on vblank or display flip) • Render engine goes IDLE due to head point equal to tail pointer

24:17 **Reserved**

Format:	MBZ

16:0 **Surface Pitch**

Format:	U17-1 Pitch in Bytes

This field specifies the pitch of the hierarchical depth buffer in (#Bytes - 1).

Value	Name
[127, 3FFFFh]	corresponding to [128B, 128KB] also restricted to a multiple of 128B

Programming Notes
Since this surface is tiled, the pitch specified must be a multiple of the tile pitch, in the range [128B, 128KB].

2 31:0 **Surface Base Address**

Format:	GraphicsAddress[31:0]HierarchicalDepthBuffer

This field specifies the starting Dword address of the buffer in mapped Graphics Memory.

Programming Notes
The Hierarchical Depth Buffer can only be mapped to Main Memory (uncached).

3DSTATE_HS

<table>
<tr><td colspan="3">Source: RenderCS
Length Bias: 2</td></tr>
<tr><td colspan="3">Controls the HS stage hardware.</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h GFXPIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Command SubType</b><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></td><td>26:24</td><td><b>3D Command Opcode</b><table><tr><td>Default Value:</td><td>0h 3DSTATE_PIPELINED</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>23:16</td><td><b>3D Command Sub Opcode</b><table><tr><td>Default Value:</td><td>1Bh 3DSTATE_HS</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Format:</td><td>=n</td></tr></table><table><tr><th>Value</th><th>Name</th></tr><tr><td>5</td><td>Excludes DWord (0,1) <b>[Default]</b></td></tr></table></td></tr>
<tr><td>1</td><td>31:30</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>29:27</td><td><b>Sampler Count</b><table><tr><td>Format:</td><td>U3</td></tr></table>Specifies how many samplers (in multiples of 4) the HS kernels use. Used only for prefetching the associated sampler state entries.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>No Samplers</td><td>no samplers used</td></tr><tr><td>1h</td><td>1-4 Samplers</td><td>between 1 and 4 samplers used</td></tr><tr><td>2h</td><td>5-8 Samplers</td><td>between 5 and 8 samplers used</td></tr><tr><td>3h</td><td>9-12 Samplers</td><td>between 9 and 12 samplers used</td></tr><tr><td>4h</td><td>13-16 Samplers</td><td>between 13 and 16 samplers used</td></tr><tr><td>5h-7h</td><td>Reserved</td><td>Reserved</td></tr></table></td></tr>
<tr><td></td><td>26</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
</table>

<table>
<tr><th colspan="3">3DSTATE_HS</th></tr>
<tr><td></td><td>25:18</td><td><b>Binding Table Entry Count</b>
<table><tr><td>Format:</td><td>U8</td></tr></table>
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.</td></tr>
<tr><td></td><td>17</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>16</td><td><b>Floating Point Mode</b>
Specifies the initial floating point mode used by the dispatched thread.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>IEEE-754</td><td>Use IEEE-754 Rules</td></tr>
<tr><td>1h</td><td>alternate</td><td>Use alternate rules</td></tr>
</table></td></tr>
<tr><td></td><td>15:14</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>13</td><td><b>Illegal Opcode Exception Enable</b>
<table><tr><td>Format:</td><td>Enable</td></tr></table>
This bit gets loaded into EU CR0.1[12] (note the bit # difference). See Exceptions and ISA Execution Environment.</td></tr>
<tr><td></td><td>12</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11:8</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7</td><td><b>Software Exception Enable</b>
<table><tr><td>Format:</td><td>Enable</td></tr></table>
This bit gets loaded into EU CR0.1[13] (note the bit # difference). See Exceptions and ISA Execution Environment.</td></tr>
<tr><td></td><td>6:0</td><td><b>Maximum Number of Threads</b>
<table><tr><td>Format:</td><td>U7-1 thread count</td></tr></table>
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. Limit is based on max number of HS URB handles.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>[0,15]</td><td>indicating a thread count of [1,16]</td></tr>
</table></td></tr>
</table>

<table>
<tr><th colspan="3">3DSTATE_HS</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>A URB_FENCE command must be issued subsequent to any change to the value in this field and before any subsequent pipeline processing (e.g., via 3DPRIMITIVE or CONSTANT_BUFFER). See Graphics Processing Engine (Command Ordering Rules)</td></tr>
</table>
</td></tr>
<tr><td>2</td><td>31</td><td>
<b>Enable</b>
<table>
<tr><td>Format:</td><td>Enable</td></tr>
</table>
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.
<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>
</td></tr>
<tr><td></td><td>30</td><td>
<b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table>
</td></tr>
<tr><td></td><td>29</td><td>
<b>Statistics Enable</b>
<table>
<tr><td>Format:</td><td>Enable</td></tr>
</table>
This bit controls whether HS-unit-specific statistics register(s) will increment (for each patch).
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Disable</td><td>HS_INVOCATIONS_COUNT will not increment</td></tr>
<tr><td>1h</td><td>Enable</td><td>HS_INVOCATIONS_COUNT will increment</td></tr>
</table>
</td></tr>
<tr><td></td><td>28:18</td><td>
<b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table>
</td></tr>
<tr><td></td><td>17:8</td><td>
<b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table>
</td></tr>
<tr><td></td><td>7:4</td><td>
<b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table>
</td></tr>
<tr><td></td><td>3:0</td><td>
<b>Instance Count</b>
<table>
<tr><td>Format:</td><td>U4-1</td></tr>
</table>
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 <b>Instance Count</b> to the number of threads that can be simultaneously active within a half-slice. Factors which must be considered includes scratch memory availability.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>[0,15]</td><td></td><td>representing [1,16] instances</td></tr>
</table>
</td></tr>
<tr><td>3</td><td>31:6</td><td>
<b>Kernel Start Pointer</b>
<table>
<tr><td>Format:</td><td>InstructionBaseOffset[31:6]Kernel</td></tr>
</table>
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
</td></tr>
</table>

<table>
<tr><th colspan="3">3DSTATE_HS</th></tr>
<tr><td></td><td></td><td>Base Address.</td></tr>
<tr><td></td><td>5:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>4</td><td>31:10</td><td><b>Scratch Space Base Pointer</b><table><tr><td>Format:</td><td>GeneralStateOffset[31:10]</td></tr></table>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.</td></tr>
<tr><td></td><td>9:4</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>3:0</td><td><b>Per-Thread Scratch Space</b><table><tr><td>Format:</td><td>U4 power of 2 Bytes over 1K Bytes</td></tr></table>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.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,11]</td><td>indicating [1K Bytes, 2M Bytes]</td></tr></table></td></tr>
<tr><td>5</td><td>31:28</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>27</td><td><b>Single Program Flow</b>
 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.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Reserved</td><td></td></tr><tr><td>1h</td><td>Enable</td><td>Single Program Flow enabled</td></tr></table></td></tr>
<tr><td></td><td>26</td><td><b>Vector Mask Enable</b><table><tr><td>Format:</td><td>U1 FormatDesc: Enumerated Type</td></tr></table>When SPF=0, 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.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Dmask</td><td>Channels are enabled based on the dispatch mask</td></tr><tr><td>1h</td><td>Vmask</td><td>Channels are enabled based on the vector mask</td></tr></table></td></tr>
<tr><td></td><td>25</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>24</td><td><b>Include Vertex Handles</b><table><tr><td>Format:</td><td>Boolean</td></tr></table>If set, all the input Vertex URB handles are included in payloads. This field is ignored if HS</td></tr>
</table>

3DSTATE_HS

<table>
<tr><td></td><td></td><td>Function Enable is DISABLED. Programming Restriction:This field must be set if value if Vertex URB Entry Read Length is cleared to zero.</td></tr>
<tr><td></td><td>23:19</td><td><b>Dispatch GRF Start Register For URB Data</b><table><tr><td>Format:</td><td>U5</td></tr></table>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.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,31]</td><td>indicating GRF [R0,R31]</td></tr></table></td></tr>
<tr><td></td><td>18:17</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>16:11</td><td><b>Vertex URB Entry Read Length</b><table><tr><td>Format:</td><td>U6</td></tr></table>Specifies the amount of URB data read and passed in the thread payload for each Vertex URB entry, in 256-bit register increments. This field is ignored if HS Function Enable is DISABLED. Programming Restriction:This field must be a non-zero value if Include Vertex Handles is cleared to zero.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,63]</td><td></td></tr></table></td></tr>
<tr><td></td><td>10</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>9:4</td><td><b>Vertex URB Entry Read Offset</b><table><tr><td>Format:</td><td>U6</td></tr></table>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.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,63]</td><td></td></tr></table></td></tr>
<tr><td></td><td>3:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>6</td><td>31:16</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>15:13</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>12</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11:0</td><td><b>Semaphore Handle</b><table><tr><td>Format:</td><td>URBOffset[17:6]</td></tr></table>This is the URB offset pointing to the first of the GS semaphore DWords in the URB. The size of the region is 32 DWs(16 - 512b URB entries). Software is responsible for allocating combined GS and/or HS semaphore Dwords in a single contiguous region of the URB. Software must also</td></tr>
</table>

3DSTATE_HS		
		make sure the 3D pipeline is IDLE prior to allocating or deallocating the region. The semaphores can be located in an unused area within a FF unit's URB fenced region or an unused area within the Push Constant region.

3DSTATE_HS

Source: RenderCS

Length Bias: 2

Controls the HS stage hardware.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h GFXPIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Command SubType</b><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></td><td>26:24</td><td><b>3D Command Opcode</b><table><tr><td>Default Value:</td><td>0h 3DSTATE_PIPELINED</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>23:16</td><td><b>3D Command Sub Opcode</b><table><tr><td>Default Value:</td><td>1Bh 3DSTATE_HS</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>5 Excludes DWord (0,1)</td></tr><tr><td>Format:</td><td>=n</td></tr></table></td></tr>
<tr><td>1</td><td>31:30</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>29:27</td><td><b>Sampler Count</b><table><tr><td>Format:</td><td>U3</td></tr></table>Specifies how many samplers (in multiples of 4) the HS kernels use. Used only for prefetching the associated sampler state entries.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>No Samplers</td><td>no samplers used</td></tr><tr><td>1h</td><td>1-4 Samplers</td><td>between 1 and 4 samplers used</td></tr><tr><td>2h</td><td>5-8 Samplers</td><td>between 5 and 8 samplers used</td></tr><tr><td>3h</td><td>9-12 Samplers</td><td>between 9 and 12 samplers used</td></tr><tr><td>4h</td><td>13-16 Samplers</td><td>between 13 and 16 samplers used</td></tr><tr><td>5h-7h</td><td>Reserved</td><td></td></tr></table></td></tr>
<tr><td></td><td>26</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>25:18</td><td><b>Binding Table Entry Count</b><table><tr><td>Format:</td><td>U8</td></tr></table></td></tr>
</table>

3DSTATE_HS

<table>
<tr><td></td><td></td><td>When <b>HW Generated Binding Table</b> is disabled:
 Specifies how many binding table entries the kernel uses. Used only for prefetching of the binding table entries and associated surface state.<table><tr><th>Programming Notes</th></tr><tr><td>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.</td></tr></table></td></tr>
<tr><td></td><td>17</td><td><b>Thread Priority</b>
Specifies the priority of the thread for dispatch<table><tr><th>Value</th><th>Name</th></tr><tr><td>0h</td><td>Normal Priority</td></tr><tr><td>1h</td><td>High Priority</td></tr></table></td></tr>
<tr><td></td><td>16</td><td><b>Floating Point Mode</b>
Specifies the initial floating point mode used by the dispatched thread.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>IEEE-754</td><td>Use IEEE-754 Rules</td></tr><tr><td>1h</td><td>alternate</td><td>Use alternate rules</td></tr></table></td></tr>
<tr><td></td><td>15:14</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>13</td><td><b>Illegal Opcode Exception Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>This bit gets loaded into EU CR0.1[12] (note the bit # difference). See <i>Exceptions</i> and <i>ISA Execution Environment</i>.</td></tr>
<tr><td></td><td>12</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7</td><td><b>Software Exception Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>This bit gets loaded into EU CR0.1[13] (note the bit # difference). See <i>Exceptions</i> and <i>ISA Execution Environment</i>.</td></tr>
<tr><td></td><td>6:0</td><td><b>Maximum Number of Threads</b><table><tr><td>Format:</td><td>U7-1 Thread count</td></tr></table>Specifies the maximum number of simultaneous threads allowed to be active. Used to avoid using up the scratch space, or to avoid potential deadlock. Limit is based on max number of HS URB handles.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,15]</td><td>indicating thread count of [1,16]</td></tr></table><table><tr><th>Programming Notes</th></tr></table></td></tr>
</table>

<table>
<tr><th colspan="3">3DSTATE_HS</th></tr>
<tr><td></td><td></td><td><table><tr><td>A URB_FENCE command must be issued subsequent to any change to the value in this field and before any subsequent pipeline processing (e.g., via 3DPRIMITIVE or CONSTANT_BUFFER). See <i>Graphics Processing Engine</i> (Command Ordering Rules)</td></tr></table></td></tr>
<tr><td>2</td><td>31</td><td><b>HS Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>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.<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></td></tr>
<tr><td></td><td>30</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>29</td><td><b>HS Statistics Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>This bit controls whether HS-unit-specific statistics register(s) will increment (for each patch).<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Disable</td><td>HS_INVOCATIONS_COUNT will not increment</td></tr><tr><td>1h</td><td>Enable</td><td>HS_INVOCATIONS_COUNT will increment</td></tr></table></td></tr>
<tr><td></td><td>28:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>Instance Count</b><table><tr><td>Format:</td><td>U8</td></tr></table>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 <b>Instance Count</b> to the number of threads that can be simultaneously active within a half-slice. Factors which must be considered includes scratch memory availability.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,15]</td><td>representing [1,16] instances</td></tr></table></td></tr>
<tr><td>3</td><td>31:6</td><td><b>Kernel Start Pointer</b><table><tr><td>Format:</td><td>InstructionBaseOffset[31:6]Kernel</td></tr></table>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 <b>Instruction Base Address</b>.</td></tr>
<tr><td></td><td>5:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>4</td><td>31:10</td><td><b>Scratch Space Base Pointer</b><table><tr><td>Format:</td><td>GeneralStateOffset[31:10]</td></tr></table></td></tr>
</table>

<table>
<tr><th colspan="3">3DSTATE_HS</th></tr>
<tr><td></td><td></td><td>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.</td></tr>
<tr><td></td><td>9:4</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>3:0</td><td><b>Per-Thread Scratch Space</b><table><tr><td>Format:</td><td>U4 power of 2 Bytes over 1K Bytes</td></tr></table>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.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>[0,11]</td><td></td><td>indicating [1K Bytes, 2M Bytes]</td></tr></table></td></tr>
<tr><td>5</td><td>31:28</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>27</td><td><b>Single Program Flow (SPF)</b>
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 <i>ISA Execution Environment</i>.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Reserved</td><td></td></tr><tr><td>1h</td><td>Enable</td><td>Single Program Flow enabled</td></tr></table></td></tr>
<tr><td></td><td>26</td><td><b>Vector Mask Enable (VME)</b><table><tr><td>Format:</td><td>U1 Enumerated Type</td></tr></table>When SPF=0, 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.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Dmask</td><td>Channels are enabled based on the dispatch mask</td></tr><tr><td>1h</td><td>Vmask</td><td>Channels are enabled based on the vector mask</td></tr></table></td></tr>
<tr><td></td><td>25</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>24</td><td><b>Include Vertex Handles</b><table><tr><td>Format:</td><td>Boolean</td></tr></table>If set, all the input Vertex URB handles are included in payloads.
This field is ignored if <b>HS Function Enable</b> is DISABLED.<table><tr><th>Programming Notes</th></tr><tr><td>This field must be set if value if <b>Vertex URB Entry Read Length</b> is cleared to zero.</td></tr></table></td></tr>
<tr><td></td><td>23:19</td><td><b>Dispatch GRF Start Register for URB Data</b><table><tr><td>Format:</td><td>U5</td></tr></table>Specifies the starting GRF register number for the URB portion (Constant + Vertices) of the</td></tr>
</table>

3DSTATE_HS

<table>
<tr><td></td><td></td><td>thread payload.
This field is ignored if <b>HS Function Enable</b> is DISABLED.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,31]</td><td>indicating GRF [R0,R31]</td></tr></table></td></tr>
<tr><td></td><td>18:17</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>16:11</td><td><b>Vertex URB Entry Read Length</b>
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.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,63]</td><td></td></tr></table><table><tr><th>Programming Notes</th></tr><tr><td>This field must be a non-zero value if <b>Include Vertex Handles</b> is cleared to zero.</td></tr></table></td></tr>
<tr><td></td><td>10</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>9:4</td><td><b>Vertex URB Entry Read Offset</b>
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.</td></tr>
<tr><td></td><td>3:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>6</td><td>31:12</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11:0</td><td><b>Semaphore Handle</b><table><tr><td>Format:</td><td>U12 Handle</td></tr></table>This is the 512b-aligned URB handle pointing to the first of the 32 HS semaphore DWords in the URB.
Software is responsible for statically allocating combined HS and/or GS semaphore Dwords in a single contiguous region of the URB. The semaphores can be located in an unused area within a FF unit's URB fenced region or an unused area within the Push Constant region.
This field is ignored if <b>HS Function Enable</b> is DISABLED.</td></tr>
</table>

3DSTATE_INDEX_BUFFER

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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h GFXPIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Command SubType</b><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></td><td>26:24</td><td><b>3D Command Opcode</b><table><tr><td>Default Value:</td><td>0h 3DSTATE_PIPELINED</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>23:16</td><td><b>3D Command Sub Opcode</b><table><tr><td>Default Value:</td><td>0Ah 3DSTATE_INDEX_BUFFER</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:12</td><td><b>Index Buffer Memory Object Control State</b><table><tr><td>Format:</td><td>MEMORY_OBJECT_CONTROL_STATE</td></tr></table>Specifies the memory object control state for this index buffer.</td></tr>
<tr><td></td><td>11</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>10</td><td><b>Cut Index Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table><table><tr><td>If ENABLED, the largest index value (0xFF,0xFFFF,0xFFFFFFFF, depending on Index Format) is interpreted as the "cut" index. (See description of this elsewhere in this section).</td></tr><tr><td>(Expected OpenGL driver usage). This field can only be enabled for certain primitive topology types. Refer to the table later in this section for details.</td></tr></table></td></tr>
<tr><td></td><td>9:8</td><td><b>Index Format</b><table><tr><td>Format:</td><td>U2 enumerated type</td></tr></table>This field specifies the data format of the index buffer. All index values are UNSIGNED.<table><tr><th>Value</th><th>Name</th></tr><tr><td>0h</td><td>INDEX_BYTE</td></tr></table></td></tr>
</table>

<table>
<tr><th colspan="3">3DSTATE_INDEX_BUFFER</th></tr>
<tr><td></td><td></td><td><table><tr><td>1h</td><td>INDEX_WORD</td></tr><tr><td>2h</td><td>INDEX_DWORD</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td>DWord Length<table><tr><td>Default Value:</td><td>1h Excludes DWord (0,1)</td></tr><tr><td>Format:</td><td>=n Total Length - 2</td></tr></table></td></tr>
<tr><td>1</td><td>31:0</td><td>Buffer Starting Address<table><tr><td>Format:</td><td>GraphicsAddress[31:0]Index_Buffer_Entry</td></tr></table><table><tr><td>This field contains the size-aligned (as specified by Index Format) Graphics Address 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.</td></tr></table><table><tr><th>Programming Notes</th></tr><tr><td>Index Buffers can only be allocated in linear (not tiled) graphics memory.</td></tr></table></td></tr>
<tr><td>2</td><td>31:0</td><td>Buffer Ending Address<table><tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr></table>If non-zero, this field contains the address of the last valid byte in the index buffer. Any index buffer reads past this address returns an index value of 0 (as if the index buffer was zero-extended). Software must guarantee that the buffer ends on an index boundary (e.g., for an INDEX_DWORD buffer, Bits [1:0] == 11b).</td></tr>
</table>

3DSTATE_LINE_STIPPLE

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** Format: MBZ
	7:0	**Dword Length** Default Value: 1h Excludes Dword (0,1) Format: =n Total Length - 2
1	31	**Modify Enable (Current Repeat Counter, Current Stipple Index)** Format: Enable Modify enable for **Current Repeat Counter** and **Current Stipple Index** fields. **Programming Notes** Software should never set this field to enabled. It is provided only for HW-generated commands as part of context save/restore.
	30	**Reserved** Format: MBZ
	29:21	**Current Repeat Counter** Format: U9 This field sets the HW-internal repeat counter state. Note: Software should never attempt to set this value - this state is only provided for HW-generated commands as part of context save/restore.
	20	**Reserved** Format: MBZ
	19:16	**Current Stipple Index** Format: U4

3DSTATE_LINE_STIPPLE

		This field sets the HW-internal stipple pattern index. Note: Software should never attempt to set this value - this state is only provided for HW-generated commands as part of context save/restore.
	15:0	**Line Stipple Pattern** Format: | 16 bit mask Bit 15 = most significant bit, Bit 0 = least significant bit Specifies a pattern used to mask out bit specific pixels while rendering lines.
2	31:15	**Line Stipple Inverse Repeat Count** Format: | U1.16 Range: [0.00390625, 1.0] Specifies the inverse (truncated) of the repeat count for the line stipple function.
	14:9	**Reserved** Format: | MBZ
	8:0	**Line Stipple Repeat Count** Format: | U9 Specifies the repeat count for the line stipple function. **Value** | **Name** [1, 256] |

3DSTATE_MONOFILTER_SIZE

Source:	RenderCS
Length Bias:	2

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

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h GFXPIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Command SubType</b><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></td><td>26:24</td><td><b>3D Command Opcode</b><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></td><td>23:16</td><td><b>3D Command Sub Opcode</b><table><tr><td>Default Value:</td><td>11h 3DSTATE_MONOFILTER_SIZE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>0h Excludes DWord (0,1)</td></tr><tr><td>Format:</td><td>=n</td></tr></table>Total Length - 2</td></tr>
<tr><td>1</td><td>31:6</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>5:3</td><td><b>Monochrome Filter Width</b><table><tr><td>Format:</td><td>U3</td></tr></table>This field specifies the width of the monochrome filter. It is ignored if the monochrome filter is not enabled.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[1,7]</td><td></td></tr></table></td></tr>
<tr><td></td><td>2:0</td><td><b>Monochrome Filter Height</b><table><tr><td>Format:</td><td>U3</td></tr></table>This field specifies the height of the monochrome filter. It is ignored if the monochrome filter is not enabled.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[1,7]</td><td></td></tr></table></td></tr>
</table>

3DSTATE_MULTISAMPLE

Source:	RenderCS
Length Bias:	2

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

Programming Restriction:

Driver must ierarchi that all the caches in the depth pipe are flushed before this command is parsed. This requires driver to send a PIPE_CONTROL with a CS stall along with a Depth Flush prior to this command.

When this command is issued, the currently active depth buffer, hierarchical depth buffer, stencil buffer, and render target(s) must be cleared (meaning that every pixel must be overwritten). Alternatively, other surfaces can be activated before issuing the next 3DPRIMITIVE that were previously rendered with the same values of all state fields in this command. In other words, it is illegal to render to these surfaces with multiple different values of the state fields in this command.

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

3DSTATE_MULTISAMPLE

5 **Reserved**

Format:	MBZ

4 **Pixel Location**

Format:	U1

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

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

Programming Notes
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.

3:1 **Number of Multisamples**

Format:	U3 enumerated value

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
0h	NUMSAMPLES_1	1 sample/pixel
1h	Reserved	
2h	NUMSAMPLES_4	4 samples/pixel
3h	NUMSAMPLES_8	8 samples/pixel
[4h,7h]	Reserved	

Programming Notes
Setting **Multisample Rasterization Mode** to MSRASTMODE_xxx_PATTERN when **Number of Multisamples** == NUMSAMPLES_1 is UNDEFINED. The setting of this field must match the **Number of Multisamples** field in SURFACE_STATE of all bound render targets.

0 **Reserved**

Format:	MBZ

2 31:28 **Sample3 X Offset**

Format:	U0.4

Description
Subpixel X offset of Sample 3 relative to the UL pixel origin. Valid only when NUMSAMPLES_4 or _8. Setting ignored when not in MSRASTMODE_xxx_PATTERN mode.
Valid when NUMSAMPLES_1

3DSTATE_MULTISAMPLE

Value	Name
[0,15]	[0,0.9375]

27:24 **Sample3 Y Offset**

Format:	U0.4

Description
Subpixel Y offset of Sample 3 relative to the UL pixel origin. Valid only when NUMSAMPLES_4 or _8. Setting ignored when not in MSRASTMODE_xxx_PATTERN mode.
Valid when NUMSAMPLES_1

Value	Name
[0,15]	[0,0.9375]

23:20 **Sample2 X Offset**

Format:	U0.4

Description
Subpixel X offset of Sample 2 relative to the UL pixel origin. Valid only when NUMSAMPLES_4 or _8. Setting ignored when not in MSRASTMODE_xxx_PATTERN mode.
Valid when NUMSAMPLES_1

Value	Name
[0,15]	[0,0.9375]

19:16 **Sample2 Y Offset**

Format:	U0.4

Description
Subpixel Y offset of Sample 2 relative to the UL pixel origin. Valid only when NUMSAMPLES_4 or _8. Setting ignored when not in MSRASTMODE_xxx_PATTERN mode.
Valid when NUMSAMPLES_1

Value	Name
[0,15]	[0,0.9375]

15:12 **Sample1 X Offset**

Format:	U0.4

Description
Subpixel X offset of Sample 1 relative to the UL pixel origin. Valid only when NUMSAMPLES_4 or _8. Setting ignored when not in MSRASTMODE_xxx_PATTERN

3DSTATE_MULTISAMPLE

mode.
Valid when NUMSAMPLES_1

Value	Name
[0,15]	[0,0.9375]

11:8 **Sample1 Y Offset**

Format:	U0.4

Description
Subpixel Y offset of Sample 1 relative to the UL pixel origin. Valid only when NUMSAMPLES_4 or _8. Setting ignored when not in MSRASTMODE_xxx_PATTERN mode.
Valid when NUMSAMPLES_1

Value	Name
[0,15]	[0,0.9375]

7:4 **Sample0 X Offset**

Format:	U0.4

Description
Subpixel X offset of Sample 0 relative to the UL pixel origin. Valid only when NUMSAMPLES_4 or _8. Setting ignored when not in MSRASTMODE_xxx_PATTERN mode.
Valid when NUMSAMPLES_1

Value	Name
[0,15]	[0,0.9375]

3:0 **Sample0 Y Offset**

Format:	U0.4

Description
Subpixel Y offset of Sample 0 relative to the UL pixel origin. Valid only when NUMSAMPLES_4 or _8. Setting ignored when not in MSRASTMODE_xxx_PATTERN mode.
Valid when NUMSAMPLES_1

Value	Name
[0,15]	[0,0.9375]

3 31:28 **Sample7 X Offset**

Format:	U0.4

Subpixel X offset of Sample 7 relative to the UL pixel origin. Valid only when NUMSAMPLES_8.

<table>
<tr><th colspan="3">3DSTATE_MULTISAMPLE</th></tr>
<tr><td></td><td></td><td>Setting ignored when not in MSRASTMODE_xxx_PATTERN mode.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,15]</td><td>[0,0.9375]</td></tr></table></td></tr>
<tr><td></td><td>27:24</td><td><b>Sample7 Y Offset</b><table><tr><td>Format:</td><td>U0.4</td></tr></table>Subpixel Y offset of Sample 7 relative to the UL pixel origin. Valid only when NUMSAMPLES_8. Setting ignored when not in MSRASTMODE_xxx_PATTERN mode.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,15]</td><td>[0,0.9375]</td></tr></table></td></tr>
<tr><td></td><td>23:20</td><td><b>Sample6 X Offset</b><table><tr><td>Format:</td><td>U0.4</td></tr></table>Subpixel X offset of Sample 6 relative to the UL pixel origin. Valid only when NUMSAMPLES_8. Setting ignored when not in MSRASTMODE_xxx_PATTERN mode.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,15]</td><td>[0,0.9375]</td></tr></table></td></tr>
<tr><td></td><td>19:16</td><td><b>Sample6 Y Offset</b><table><tr><td>Format:</td><td>U0.4</td></tr></table>Subpixel Y offset of Sample 6 relative to the UL pixel origin. Valid only when NUMSAMPLES_8. Setting ignored when not in MSRASTMODE_xxx_PATTERN mode.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,15]</td><td>[0,0.9375]</td></tr></table></td></tr>
<tr><td></td><td>15:12</td><td><b>Sample5 X Offset</b><table><tr><td>Format:</td><td>U0.4</td></tr></table>Subpixel X offset of Sample 5 relative to the UL pixel origin. Valid only when NUMSAMPLES_8. Setting ignored when not in MSRASTMODE_xxx_PATTERN mode.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,15]</td><td>[0,0.9375]</td></tr></table></td></tr>
<tr><td></td><td>11:8</td><td><b>Sample5 Y Offset</b><table><tr><td>Format:</td><td>U0.4</td></tr></table>Subpixel Y offset of Sample 5 relative to the UL pixel origin. Valid only when NUMSAMPLES_8. Setting ignored when not in MSRASTMODE_xxx_PATTERN mode.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,15]</td><td>[0,0.9375]</td></tr></table></td></tr>
<tr><td></td><td>7:4</td><td><b>Sample4 X Offset</b><table><tr><td>Format:</td><td>U0.4</td></tr></table>Subpixel X offset of Sample 4 relative to the UL pixel origin. Valid only when NUMSAMPLES_8. Setting ignored when not in MSRASTMODE_xxx_PATTERN mode.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,15]</td><td>[0,0.9375]</td></tr></table></td></tr>
<tr><td></td><td>3:0</td><td><b>Sample4 Y Offset</b></td></tr>
</table>

3DSTATE_MULTISAMPLE

Format:	U0.4

Subpixel Y offset of Sample 4 relative to the UL pixel origin. Valid only when NUMSAMPLES_8. Setting ignored when not in MSRASTMODE_xxx_PATTERN mode.

Value	Name
[0,15]	[0,0.9375]

3DSTATE_POLY_STIPPLE_OFFSET

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** Format: MBZ
	7:0	**Dword Length** Default Value: 0h Excludes Dword (0,1) Format: =n Total Length - 2
1	31:13	**Reserved** Format: MBZ
	12:8	**Polygon Stipple X Offset** Format: U5 Specifies a 5 bit x address offset in the poly stipple pattern Value: [0,31] Name:
	7:5	**Reserved** Format: MBZ
	4:0	**Polygon Stipple Y Offset** Format: U5 Specifies a 5 bit y address offset in the poly stipple pattern Value: [0,31] Name:

3DSTATE_POLY_STIPPLE_PATTERN

Source: RenderCS
Length Bias: 2

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.

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: 07h 3DSTATE_POLY_STIPPLE_PATTERN Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**Dword Length** Default Value: 1Fh Excludes Dword (0,1) Format: =n Total Length - 2
1	31:0	**Polygon Stipple Pattern Row 1 (top most)** Format: 32 bit mask Bit 31 = upper left corner, Bit 0 = upper right corner of first row. Specifies a pattern used by Polygon Stipple to mask out specific pixels of every 32x32 area rendered.
2..32	31:0	**Polygon Stipple Pattern Rows 2-32 (bottom most)** Format: 32 bit mask Bit 31 = upper left corner, Bit 0 = upper right corner of first row. Specifies a pattern used by Polygon Stipple to mask out specific pixels of every 32x32 area rendered.

3DSTATE_PS

Source: RenderCS
Length Bias: 2

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h GFXPIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Command SubType</b><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></td><td>26:24</td><td><b>3D Command Opcode</b><table><tr><td>Default Value:</td><td>0h 3DSTATE_PIPELINED</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>23:16</td><td><b>3D Command Sub Opcode</b><table><tr><td>Default Value:</td><td>20h 3DSTATE_PS</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>06h Excludes DWord (0,1)</td></tr><tr><td>Format:</td><td>=n</td></tr></table>Total Length - 2</td></tr>
<tr><td>1</td><td>31:6</td><td><b>Kernel Start Pointer[0]</b><table><tr><td>Format:</td><td>InstructionBaseOffset[31:6]Kernel</td></tr></table>Specifies the 64-byte aligned address offset of the first instruction in the kernel[0]. This pointer is relative to the Instruction Base Address.</td></tr>
<tr><td></td><td>5:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>2</td><td>31</td><td><b>Single Program Flow (SPF)</b>
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.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Multiple</td><td>Multiple Program Flows</td></tr><tr><td>1h</td><td>Single</td><td>Single Program Flows</td></tr></table></td></tr>
<tr><td></td><td>30</td><td><b>Vector Mask Enable (VME)</b><table><tr><td>Format:</td><td>U1 Enumerated Type</td></tr></table>When SPF=0, 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.</td></tr>
</table>

3DSTATE_PS

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

29:27 **Sampler Count**

Format:	U3

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

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

26 **Denormal Mode**

Specifies the denornal mode used by the dispatched thread.

Value	Name	Description
0h	FTZ	Denormals are flushed to zero
1h	RET	Denormals are retained

25:18 **Binding Table Entry Count**

Format:	U8

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.

Value	Name
[0,255]	

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

17 **Reserved**

Format:	MBZ

16 **Floating Point Mode**

Specifies the floating point mode used by the dispatched thread.

Value	Name	Description
0h	IEEE-745	Use IEEE-754 rules
1h	Alt	Use alternate rules

<table>
<tr><th colspan="3">3DSTATE_PS</th></tr>
<tr><td></td><td>15:14</td><td><b>Rounding Mode</b>
Specifies the rounding mode used by the dispatched thread.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>RTNE</td><td>Round to Nearest Even</td></tr><tr><td>1h</td><td>RU</td><td>Round toward +infinity</td></tr><tr><td>2h</td><td>RD</td><td>Round toward -infinity</td></tr><tr><td>3h</td><td>RTZ</td><td>Round toward zero</td></tr></table></td></tr>
<tr><td></td><td>13</td><td><b>Illegal Opcode Exception Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>This bit gets loaded into EU CR0.1[12] (note the bit # difference). See Exceptions and ISA Execution Environment.</td></tr>
<tr><td></td><td>12</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11</td><td><b>Mask Stack Exception Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>This bit gets loaded into EU CR0.1[12] (note the bit # difference). See Exceptions and ISA Execution Environment.</td></tr>
<tr><td></td><td>10:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7</td><td><b>Software Exception Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>This bit gets loaded into EU CR0.1[13] (note the bit # difference). See Exceptions and ISA Execution Environment.</td></tr>
<tr><td></td><td>6:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>3</td><td>31:10</td><td><b>Scratch Space Base Pointer</b><table><tr><td>Format:</td><td>GeneralStateOffset[31:10]ScratchSpace</td></tr></table>Specifies the 1k-byte aligned address offset to scratch space for use by the kernel. This pointer is relative to the <b>General State Base Address</b>.</td></tr>
<tr><td></td><td>9:4</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>3:0</td><td><b>Per Thread Scratch Space</b><table><tr><td>Format:</td><td>U4</td></tr></table>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.</td></tr>
</table>

3DSTATE_PS

Value	Name
[0,11]	indicating [1k bytes, 2M bytes] in powers of two

4 31:24 **Maximum Number of Threads**

Format:	U8-1 representing thread count

Description
Range: WIZ Hashing Disable in GT_MODE register enabled: Range = [7,171] --> [8,172] threads. Only odd values are allowed (resulting in even max number of threads) WIZ Hashing Disable in GT_MODE register disabled: Range = [3,85] --> [4,86] threads. Only odd values are allowed (resulting in even max number of threads)
Specifies the maximum number of simultaneous threads allowed to be active. Used to avoid using up the scratch space, or to avoid potential deadlock.

Value	Name	Description
[3h,1fh]	Range	[4,32] threads

Programming Notes
If this field is changed between 3DPRIMITIVE commands, a PIPE_CONTROL command with **Stall at Pixel Scoreboard** set is required to be issued. This field must have an odd value so that the max number of PS threads is even.

23:12 **Reserved**

Format:	MBZ

11 **Push Constant Enable**

Format:	Enable

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 **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.

9 **oMask Present to RenderTarget**

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 (one or two phases) is included in Render Target Write messages. If present, the oMask data is used to mask off samples.

8 **Render Target Fast Clear Enable**

Format:	Enable

3DSTATE_PS

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 **Dual Source Blend Enable**

Format:	Enable

This field is set if dual source blend is enabled. If this bit is disabled, the data port dual source message reverts to a single source message using source 0.

6 **Render Target Resolve Enable**

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**

Format:	MBZ

4:3 **Position XY Offset Select**

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.
MSDISPMODE_PERSAMPLE is required in order to select POSOFFSET_SAMPLE.

2 **32 Pixel Dispatch Enable**

Format:	Enable

Description
Enables the Windower to dispatch 8 subspans in one payload. Note: See Note: in the table below, the Valid column indicates which products that

3DSTATE_PS

combination is supported on. Combinations of dispatch enables not listed in the table are not available on any product. A: Valid on all products B: Valid. C: Not valid. D: Valid on all products, except when in non-1x PERSAMPLE mode. E: Valid on all products, except when in PERSAMPLE mode with number of multisamples >= 8. F: Valid on all products.
Each of the three KSP values are separately specified.
In addition, each kernel has a separately-specified GRF register count.
Variable Pixel Dispatch Section: Pixel Grouping (Dispatch size) control for valid pixel dispatch combinations.

1 **16 Pixel Dispatch Enable**

Format:	Enable

Description
Enables the Windower to dispatch 4 subspans in one payload. Note: See Note: in the table below, the Valid column indicates which products that combination is supported on. Combinations of dispatch enables not listed in the table are not available on any product. A: Valid on all products B: Valid. C: Not valid. D: Valid on all products, except when in non-1x PERSAMPLE mode. E: Valid on all products, except when in PERSAMPLE mode with number of multisamples >= 8. F: Valid on all products.
Each of the three KSP values are separately specified.
In addition, each kernel has a separately-specified GRF register count.
Variable Pixel Dispatch Section: Pixel Grouping (Dispatch size) control for valid pixel dispatch combinations.

0 **8 Pixel Dispatch Enable**

Format:	Enable

Description
Enables the Windower to dispatch 2 subspans in one payload. Note: See Note: in the table below, the Valid column indicates which products that combination is supported on. Combinations of dispatch enables not listed in the table are not available on any product. A: Valid on all products B: Valid.

3DSTATE_PS

		C: Not valid. D: Valid on all products, except when in non-1x PERSAMPLE mode. E: Valid on all products, except when in PERSAMPLE mode with number of multisamples >= 8. F: Valid on all products. Each of the three KSP values are separately specified. In addition, each kernel has a separately-specified GRF register count. Variable Pixel Dispatch Section: Pixel Grouping (Dispatch size) control for valid pixel dispatch combinations.
5	31:23	**Reserved** 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** Format: MBZ
	14:8	**Dispatch GRF Start Register for Constant/Setup Data [1]** 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** Format: MBZ
	6:0	**Dispatch GRF Start Register for Constant/Setup Data [2]** Format: U7 Specifies the starting GRF register number for the Constant/Setup portion of the thread payload for kernel[2]. **Value** | **Name** [0,127] |
6	31:6	**Kernel Start Pointer[1]** Format: InstructionBaseOffset[31:6]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** Format: MBZ

3DSTATE_PS		
7	31:6	**Kernel Start Pointer[2]** Format: InstructionBaseOffset[31:6]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** Format: MBZ

3DSTATE_PUSH_CONSTANT_ALLOC_DS

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	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** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:20	**Reserved** Format: MBZ
	19:16	**Constant Buffer Offset** Format: U4 Specifies the offset of the DS constant buffer into the URB. Value: [0,15] — Name: (0KB - 15KB)

3DSTATE_PUSH_CONSTANT_ALLOC_DS

0h	0KB **[Default]**

15:5 **Reserved**

Format:	MBZ

4:0 **Constant Buffer Size**

Format:	U5

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.

Value	Name
[0,15]	(0KB - 15KB) Increments of 1KB
0h	0KB **[Default]**

3DSTATE_PUSH_CONSTANT_ALLOC_GS

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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h GFXPIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Command SubType</b><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></td><td>26:24</td><td><b>3D Command Opcode</b><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></td><td>23:16</td><td><b>3D Command Sub Opcode</b><table><tr><td>Default Value:</td><td>15h 3DSTATE_PUSH_CONSTANT_ALLOC_GS</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Format:</td><td>=n</td></tr></table>Total Length - 2<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>3DSTATE_PUSH_CONSTANT_ALLOC_GS <b>[Default]</b></td><td>Excludes DWord (0,1)</td></tr></table></td></tr>
<tr><td>1</td><td>31:20</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>19:16</td><td><b>Constant Buffer Offset</b><table><tr><td>Format:</td><td>U4</td></tr></table>Specifies the offset of the GS constant buffer into the URB.<table><tr><th>Value</th><th>Name</th></tr></table></td></tr>
</table>

3DSTATE_PUSH_CONSTANT_ALLOC_GS

		[0,15]	(0KB - 15KB)
		0h	0KB **[Default]**
	15:5	**Reserved**	
		Format:	MBZ
	4:0	**Constant Buffer Size**	
		Format:	U5
		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	**Name**
		[0,15]	(0KB - 15KB) Increments of 1KB
		0h	0KB **[Default]**

3DSTATE_PUSH_CONSTANT_ALLOC_HS

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** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:20	**Reserved** Format: MBZ
	19:16	**Constant Buffer Offset** Format: U4 Specifies the offset of the HS constant buffer into the URB. Value: Name [0,15]: (0KB - 15KB)

<table>
<tr><th colspan="3">3DSTATE_PUSH_CONSTANT_ALLOC_HS</th></tr>
<tr><td></td><td></td><td><table><tr><td>0h</td><td>0KB <b>[Default]</b></td></tr></table></td></tr>
<tr><td></td><td>15:5</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>4:0</td><td><b>Constant Buffer Size</b><table><tr><td>Format:</td><td>U5</td></tr></table>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></tr><tr><td>[0,15]</td><td>(0KB - 15KB) Increments of 1KB</td></tr><tr><td>0h</td><td>0KB <b>[Default]</b></td></tr></table></td></tr>
</table>

3DSTATE_PUSH_CONSTANT_ALLOC_PS

Source: RenderCS

Length Bias: 2

Description
This command sets up the URB configuration for PS Push Constant Buffer.
A PIPE_CONTOL command with the CS Stall bit set must be programmed in the ring after this instruction.

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** Format: MBZ
	7:0	**Dword Length** Default Value: 0h Excludes Dword (0,1) Format: =n Total Length - 2
1	31:20	**Reserved** Format: MBZ
	19:16	**Constant Buffer Offset** Format: U4

3DSTATE_PUSH_CONSTANT_ALLOC_PS

Specifies the offset of the PS constant buffer into the URB.

Value	Name
[0,15]	(0KB - 15KB)
0h	0KB **[Default]**

15:5 **Reserved**

Format:	MBZ

4:0 **Constant Buffer Size**

Format:	U5

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.

Value	Name
[0,15]	(0KB - 15KB) Increments of 1KB
0h	0KB **[Default]**

3DSTATE_PUSH_CONSTANT_ALLOC_VS

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** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:20	**Reserved** Format: MBZ
	19:16	**Constant Buffer Offset** Format: U4 Specifies the offset of the VS constant buffer into the URB. Value: [0,15] — Name: (0KB - 15KB)

3DSTATE_PUSH_CONSTANT_ALLOC_VS

		0h	0KB **[Default]**
	15:5	**Reserved**	
		Format:	MBZ
	4:0	**Constant Buffer Size**	
		Format:	U5
		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	**Name**
		[0,15]	(0KB - 15KB) Increments of 1KB
		0h	0KB **[Default]**

3DSTATE_SAMPLE_MASK

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_SAMPLE_MASK Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**Dword Length** Default Value: 0h Excludes Dword (0,1) Format: =n Total Length - 2
1	31:8	**Reserved** Format: MBZ
	7:0	**Sample Mask** Format: 8 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. **Programming Notes** • If **Number of Multisamples** is NUMSAMPLES_1, bits 7:1 of this field must be zero. • If **Number of Multisamples** is NUMSAMPLES_4, bits 7:4 of this field must be zero.

3DSTATE_SAMPLER_PALETTE_LOAD0

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 Format: Opcode
	23:16	**3D Command Sub Opcode** Default Value: 02h 3DSTATE_SAMPLER_PALETTE_LOAD0 Format: Opcode
	15:8	**Reserved** 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]** Format: U8 Alpha channel loaded into the Nth entry of the texture color palette.
	23:16	**Palette Red[0:N-1]** Format: U8 Alpha channel loaded into the Nth entry of the texture color palette.
	15:8	**Palette Green[0:N-1]** Format: U8 Alpha channel loaded into the Nth entry of the texture color palette.

3DSTATE_SAMPLER_PALETTE_LOAD0

	7:0	**Palette Blue[0:N-1]** Format: U8 Alpha channel loaded into the Nth entry of the texture color palette.

3DSTATE_SAMPLER_PALETTE_LOAD1

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** 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]** Format: U8 Alpha channel loaded into the Nth entry of the texture color palette.
	23:16	**Palette Red[0:N-1]** Format: U8 Alpha channel loaded into the Nth entry of the texture color palette.
	15:8	**Palette Green[0:N-1]** Format: U8 Alpha channel loaded into the Nth entry of the texture color palette.
	7:0	**Palette Blue[0:N-1]** Format: U8 Alpha channel loaded into the Nth entry of the texture color palette.

3DSTATE_SAMPLER_STATE_POINTERS_DS

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** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:5	**Pointer to DS Sampler State** 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** Format: MBZ

3DSTATE_SAMPLER_STATE_POINTERS_GS

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** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:5	**Pointer to GS Sampler State** 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** Format: MBZ

3DSTATE_SAMPLER_STATE_POINTERS_HS

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** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:5	**Pointer to HS Sampler State** 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** Format: MBZ

3DSTATE_SAMPLER_STATE_POINTERS_PS

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** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:5	**Pointer to PS Sampler State** 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** Format: MBZ

3DSTATE_SAMPLER_STATE_POINTERS_VS

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** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:5	**Pointer to VS Sampler State** 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** Format: MBZ

3DSTATE_SBE

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: 0Ch Excludes DWord (0,1) Format: =n Total Length - 2
1	31:29	**Reserved** Format: MBZ
	28	**Attribute Swizzle Control Mode** Format: U1 enumerated type When Attribute Swizzle Enable is ENABLED, this bit controls whether attributes 0-15 or 16-31 are subject to the following swizzle controls: • Attribute n Component Override X/Y/Z/W • Attribute n Constant Source • Attribute n Swizzle Select • Attribute n Source Attribute • Attribute n Wrap Shortest Enables Note that the Number of SF Output Attributes field specifies how many attributes are output. **Note:** This field does not impact any functions which provide separate states for all 32 attributes (e.g., Point sprite, Constant interpolation). **Value** | **Name** | **Description**

3DSTATE_SBE

0h	SWIZ_0_15	Attributes 0-15 are subject to swizzling, and attributes 16-31 are not.
1h	SWIZ_16_31	Attributes 16-31 are subject to swizzling, and attributes 0-15 are not. Only valid when 16 or more attributes are output.

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

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**

Format:	U1 enumerated type

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)

19:16 **Reserved**

Format:	MBZ

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	Name
[1,16]	

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 **Reserved**

9:4 **Vertex URB Entry Read Offset**

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

3:0 **Reserved**

<table>
<tr><th colspan="3">3DSTATE_SBE</th></tr>
<tr><td></td><td></td><td><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>2..9</td><td>31</td><td><b>Attribute [2n+1] Component Override W</b><table><tr><td>Format:</td><td>Enable</td></tr></table>If set, the W component of output Attribute 1 is overridden by the W component of the constant vector specified by ConstantSource[1].</td></tr>
<tr><td></td><td>30</td><td><b>Attribute [2n+1] Component Override Z</b><table><tr><td>Format:</td><td>Enable</td></tr></table>If set, the Z component of output Attribute 1 is overridden by the Z component of the constant vector specified by ConstantSource[1].</td></tr>
<tr><td></td><td>29</td><td><b>Attribute [2n+1] Component Override Y</b><table><tr><td>Format:</td><td>Enable</td></tr></table>If set, the Y component of output Attribute 1 is overridden by the Y component of the constant vector specified by ConstantSource[1].</td></tr>
<tr><td></td><td>28</td><td><b>Attribute [2n+1] Component Override X</b><table><tr><td>Format:</td><td>Enable</td></tr></table>If set, the X component of output Attribute 1 is overridden by the X component of the constant vector specified by ConstantSource[1].</td></tr>
<tr><td></td><td>27</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>26:25</td><td><b>Attribute [2n+1] Constant Source</b><table><tr><td>Format:</td><td>U2 enumerated type</td></tr></table>This state selects a constant vector which can be used to override individual components of Attribute 1<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>CONST_0000</td><td>Constant.xyzw = 0.0,0.0,0.0,0.0</td></tr><tr><td>1h</td><td>CONST_0001_FLOAT</td><td>Constant.xyzw = 0.0,0.0,0.0,1.0</td></tr><tr><td>2h</td><td>CONST_1111_FLOAT</td><td>Constant.xyzw = 1.0,1.0,1.0,1.0</td></tr><tr><td>3h</td><td>PRIM_ID</td><td>Constant.xyzw = PrimID (replicated)</td></tr></table></td></tr>
<tr><td></td><td>24</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>23:22</td><td><b>Attribute [2n+1] Swizzle Select</b><table><tr><td>Format:</td><td>U2 enumerated type</td></tr></table>This state, along with Attribute 1 Source Attribute, specifies the source for output Attribute 1.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>INPUTATTR</td><td>This attribute is sourced from AttrInputReg[SourceAttribute]</td></tr><tr><td>1h</td><td>INPUTATTR_FACING</td><td>If the object is front-facing, this attribute is sourced</td></tr></table></td></tr>
</table>

3DSTATE_SBE

<table>
<tr><td></td><td></td><td>
<table>
<tr><td></td><td></td><td>from AttrInputReg[SourceAttribute]. If the object is back-facing, this attribute is sourced from AttrInputReg[SourceAttribute+1].</td></tr>
<tr><td>2h</td><td>INPUTATTR_W</td><td>This attribute is sourced from AttrInputReg[SourceAttribute]. The W component is copied to the X component.</td></tr>
<tr><td>3h</td><td>INPUTATTR_FACING_W</td><td>If the object is front-facing, this attribute is sourced from AttrInputReg[SourceAttribute]. If the object is back-facing, this attribute is sourced from AttrInputReg[SourceAttribute+1]. The W component is copied to the X component.</td></tr>
</table>
</td></tr>
<tr><td></td><td>21</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table>
</td></tr>
<tr><td></td><td>20:16</td><td><b>Attribute [2n+1] Source Attribute</b>
<table><tr><td>Format:</td><td>U5</td></tr></table>
This field selects the source attribute for Attribute 1. Source attribute 0 corresponds to the first 128 bits of data indicated by Vertex URB Entry Read Offset
</td></tr>
<tr><td></td><td>15</td><td><b>Attribute [2n] Component Override W</b>
<table><tr><td>Format:</td><td>Enable</td></tr></table>
If set, the W component of output Attribute 0 is overridden by the W component of the constant vector specified by ConstantSource[1].
</td></tr>
<tr><td></td><td>14</td><td><b>Attribute [2n] Component Override Z</b>
<table><tr><td>Format:</td><td>Enable</td></tr></table>
If set, the Z component of output Attribute 0 is overridden by the Z component of the constant vector specified by ConstantSource[1].
</td></tr>
<tr><td></td><td>13</td><td><b>Attribute [2n] Component Override Y</b>
<table><tr><td>Format:</td><td>Enable</td></tr></table>
If set, the Y component of output Attribute 0 is overridden by the Y component of the constant vector specified by ConstantSource[1].
</td></tr>
<tr><td></td><td>12</td><td><b>Attribute [2n] Component Override X</b>
<table><tr><td>Format:</td><td>Enable</td></tr></table>
If set, the X component of output Attribute 0 is overridden by the X component of the constant vector specified by ConstantSource[1].
</td></tr>
<tr><td></td><td>11</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table>
</td></tr>
<tr><td></td><td>10:9</td><td><b>Attribute [2n] Constant Source</b>
<table><tr><td>Format:</td><td>U2 enumerated type</td></tr></table>
This state selects a constant vector which can be used to override individual components of
</td></tr>
</table>

3DSTATE_SBE

Attribute 0

Value	Name	Description
0h	CONST_0000	Constant.xyzw = 0.0,0.0,0.0,0.0
1h	CONST_0001_FLOAT	Constant.xyzw = 0.0,0.0,0.0,1.0
2h	CONST_1111_FLOAT	Constant.xyzw = 1.0,1.0,1.0,1.0
3h	PRIM_ID	Constant.xyzw = PrimID (replicated)

8 **Reserved**

Format:	MBZ

7:6 **Attribute [2n] Swizzle Select**

Format:	U2 enumerated type

This state, along with Attribute 0 Source Attribute, specifies the source for output Attribute 0.

Value	Name	Description
0h	INPUTATTR	This attribute is sourced from AttrInputReg[SourceAttribute]
1h	INPUTATTR_FACING	If the object is front-facing, this attribute is sourced from AttrInputReg[SourceAttribute]. If the object is back-facing, this attribute is sourced from AttrInputReg[SourceAttribute+1].
2h	INPUTATTR_W	This attribute is sourced from AttrInputReg[SourceAttribute]. The W component is copied to the X component.
3h	INPUTATTR_FACING_W	If the object is front-facing, this attribute is sourced from AttrInputReg[SourceAttribute]. If the object is back-facing, this attribute is sourced from AttrInputReg[SourceAttribute+1]. The W component is copied to the X component.

5 **Reserved**

Format:	MBZ

4:0 **Attribute [2n] Source Attribute**

Format:	U5

This field selects the source attribute for Attribute 0. Source attribute 0 corresponds to the first 128 bits of data indicated by Vertex URB Entry Read Offset

10 31:0 **Point Sprite Texture Coordinate Enable**

Format:	32-bit bitmask

Description
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

3DSTATE_SBE

		the Point Sprite Texture Coordinate Origin state bit. Bit 0 corresponds to output Attribute 0. This field must be programmed to 0 when non-point primitives are rendered.
11	31:0	**Constant Interpolation Enable[31:0]** 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.
12	31:28	**Attribute 7 WrapShortest Enables** Format: Enable[4] This state selects which components (if any) of Attribute 7 are to be interpolated in a "wrap shortest" fashion. Operation is UNDEFINED if any of these bits are set and the Constant Interpolation Enable bit associated with this attribute is set. Note that wrap-shortest interpolation is only supported for Attributes 0-15. Bit 0: WrapShortest X ComponentBit 1: WrapShortest Y ComponentBit 2: WrapShortest Z ComponentBit 3: WrapShortest W Component
	27:24	**Attribute 6 WrapShortest Enables** (See above).
	23:20	**Attribute 5 WrapShortest Enables** (See above).
	19:16	**Attribute 4 WrapShortest Enables** (See above).
	15:12	**Attribute 3 WrapShortest Enables** (See above).
	11:8	**Attribute 2 WrapShortest Enables** (See above).
	7:4	**Attribute 1 WrapShortest Enables** (See above).
	3:0	**Attribute 0 WrapShortest Enables** (See above).
13	31:28	**Attribute 15 WrapShortest Enables** Format: Enable[4] This state selects which components (if any) of Attribute 15 are to be interpolated in a "wrap shortest" fashion. Operation is UNDEFINED if any of these bits are set and the Constant Interpolation Enable bit associated with this attribute is set.Bit 0: WrapShortest X ComponentBit 1: WrapShortest Y ComponentBit 2: WrapShortest Z ComponentBit 3: WrapShortest W Component
	27:24	**Attribute 14 WrapShortest Enables** (See above).
	23:20	**Attribute 13 WrapShortest Enables** (See above).
	19:16	**Attribute 12 WrapShortest Enables**

3DSTATE_SBE

		(See above).
	15:12	**Attribute 11 WrapShortest Enables** (See above).
	11:8	**Attribute 10 WrapShortest Enables** (See above).
	7:4	**Attribute 9 WrapShortest Enables** (See above).
	3:0	**Attribute 8 WrapShortest Enables** (See above).

3DSTATE_SCISSOR_STATE_POINTERS

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** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:5	**Scissor Rect Pointer** 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** Format: MBZ

<table>
<tr><th colspan="3">3DSTATE_SF</th></tr>
<tr><td colspan="3">Source: RenderCS
Length Bias: 2</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h GFXPIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Command SubType</b><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></td><td>26:24</td><td><b>3D Command Opcode</b><table><tr><td>Default Value:</td><td>0h 3DSTATE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>23:16</td><td><b>3D Command Sub Opcode</b><table><tr><td>Default Value:</td><td>13h 3DSTATE_SF</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>5h Excludes DWord (0,1)</td></tr><tr><td>Format:</td><td>=n Total Length - 2</td></tr></table></td></tr>
<tr><td>1</td><td>31:15</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>14:12</td><td><b>Depth Buffer Surface Format</b><table><tr><td>Format:</td><td>U3 Enumerated Type</td></tr></table>Specifies the format of the depth buffer. This must exactly match the Surface Format programmed via 3DSTATE_DEPTH_BUFFER. The SF requires this information in order to compute Global Depth Bias.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>D32_FLOAT_S8X24_UINT</td><td>D32_FLOAT_S8X24_UINT</td></tr><tr><td>1h</td><td>D32_FLOAT</td><td>D32_FLOAT</td></tr><tr><td>2h</td><td>D24_UNORM_S8_UINT</td><td>D24_UNORM_S8_UINT</td></tr><tr><td>3h</td><td>D24_UNORM_X8_UINT</td><td>D24_UNORM_X8_UINT</td></tr><tr><td>4h</td><td>Reserved</td><td>Reserved</td></tr><tr><td>5h</td><td>D16_UNORM</td><td>D16_UNORM</td></tr><tr><td>6h-7h</td><td>Reserved</td><td>Reserved</td></tr></table></td></tr>
<tr><td></td><td>11</td><td><b>Legacy Global Depth Bias Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>Enables the SF to use the Global Depth Offset Constant state unmodified. If this bit is not set, the</td></tr>
</table>

3DSTATE_SF

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**

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 **Global Depth Offset Enable Solid**

Format:	Enable

Enables computation and application of Global Depth Offset for SOLID objects.

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.

8 **Global Depth Offset Enable Wireframe**

Format:	Enable

Enables computation and application of Global Depth Offset when triangles are rendered in WIREFRAME mode.

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.

7 **Global Depth Offset Enable Point**

Format:	Enable

Enables computation and application of Global Depth Offset when triangles are rendered in POINT mode.

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.

6:5 **FrontFace Fill Mode**

Format:	U2 enumerated type

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

Value	Name	Description
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.
1h	WIREFRAME	Any triangle object found to be front-facing is rendered as a series of lines along the triangle boundaries (as determined by

3DSTATE_SF

		the topology type and controlled by the vertex EdgeFlags).
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). NOTE: If the triangle is clipped, points will not be rendered at clip-inserted vertices. Point will only be rendered at original vertices (if visible).
3h	Reserved	

4:3 **BackFace Fill Mode**

Format:	U2 enumerated type

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

Value	Name	Description
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.
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).
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). NOTE: If the triangle is clipped, points will not be rendered at clip-inserted vertices. Point will only be rendered at original vertices (if visible).
3h	Reserved	

2 **Reserved**

Format:	MBZ

1 **View Transform Enable**

Format:	Enable

This bit controls the Viewport Transform function.

0 **Front Winding**

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.

2 31 **Anti-Aliasing Enable**

Format:	Enable

This field enables "alpha-based" line anti-aliasing.

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

30:29 **Cull Mode**

Format:	3D_CullMode

Controls removal (culling) of triangle objects based on orientation. The cull mode only applies to

3DSTATE_SF

triangle objects and does not apply to lines, points or rectangles.

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

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

28 **Reserved**

27:18 **Line Width**

Format:	U3.7

Range: [0.0, 7.9921875]
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).

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.

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
1h		1.0 pixels
2h		2.0 pixels
3h		4.0 pixels

15 **Reserved**

Format:	MBZ

14 **Reserved**

Format:	MBZ

13 **Reserved**

12 **Reserved**

11 **Scissor Rectangle Enable**

Format:	Enable

3DSTATE_SF

Enables operation of Scissor Rectangle.

10 **Reserved**

Format:	MBZ

9:8 **Multisample Rasterization Mode**

Format:	U2 enumerated type

This state is duplicated in 3DSTATE_WM and both must be set to the same value. See the field in 3DSTATE_WM for definition details.

7:0 **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	Vertex 0
1h	Vertex 1
2h	Vertex 2
3h	Reserved

28:27 **Line Strip/List Provoking Vertex Select**

Format:	0-based vertex index

Selects which vertex of a line (in a line strip or list primitive) is considered the "provoking vertex".

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

<table>
<tr><th colspan="3">3DSTATE_SF</th></tr>
<tr><td></td><td></td><td><table><tr><td>1h</td><td>Vertex 1</td></tr><tr><td>2h</td><td>Vertex 2</td></tr><tr><td>3h</td><td>Reserved</td></tr></table></td></tr>
<tr><td></td><td>24:15</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>14</td><td><b>AA Line Distance Mode</b><table><tr><td>Format:</td><td>U1</td></tr></table>This bit controls the distance computation for antialiased lines.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Reserved</td><td>Reserved</td></tr><tr><td>1h</td><td>AALINEDISTANCE_TRUE</td><td>True distance computation. This is the normal setting which should yield WHQL compliance.</td></tr></table></td></tr>
<tr><td></td><td>13</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>12</td><td><b>Vertex Sub Pixel Precision Select</b><table><tr><td>Format:</td><td>U1</td></tr></table>Selects the number of fractional bits maintained in the vertex data<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Disable</td><td>8 sub pixel precision bits maintained</td></tr><tr><td>1h</td><td>Enable</td><td>4 sub pixel precision bits maintained</td></tr></table></td></tr>
<tr><td></td><td>11</td><td><b>Use Point Width State</b><table><tr><td>Format:</td><td>U1</td></tr></table>Controls whether the point width passed on the vertex or from state is used for rendering point primitives.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td></td><td>Use Point Width on Vertex</td></tr><tr><td>1h</td><td></td><td>Use Point Width from State</td></tr></table></td></tr>
<tr><td></td><td>10:0</td><td><b>Point Width</b><table><tr><td>Format:</td><td>U8.3</td></tr></table><table><tr><td>Range: [0.125, 255.875] pixels</td></tr><tr><td>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</td></tr></table></td></tr>
<tr><td>4</td><td>31:0</td><td><b>Global Depth Offset Constant</b><table><tr><td>Format:</td><td>IEEE_FP</td></tr></table>Specifies the constant term in the Global Depth Offset function.</td></tr>
<tr><td>5</td><td>31:0</td><td><b>Global Depth Offset Scale</b><table><tr><td>Format:</td><td>IEEE_FP</td></tr></table>Specifies the scale term used in the Global Depth Offset function.</td></tr>
</table>

3DSTATE_SF		
6	31:0	**Global Depth Offset Clamp** Format: IEEE_FP Specifies the clamp term used in the Global Depth Offset function.

3DSTATE_SO_BUFFER

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: 18h 3DSTATE_SO_BUFFER 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	**Reserved** Format: MBZ
	30:29	**SO Buffer Index** Format: U2 Specifies which of the four SO Buffers is being defined.
	28:25	**SO Buffer Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for the SO buffer.
	24:22	**Reserved** Format: MBZ
	21:12	**Reserved** Format: MBZ
	11:0	**Surface Pitch** Format: U12 Pitch in Bytes This field specifies the pitch of the SO buffer in #Bytes.

<table>
<tr><th colspan="3">3DSTATE_SO_BUFFER</th></tr>
<tr><td></td><td></td><td><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>2</td><td>31:2</td><td><b>Surface Base Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:2]</td></tr></table>This field specifies the starting DWord address LSBs of the buffer in Graphics Memory.</td></tr>
<tr><td></td><td>1:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>3</td><td>31:2</td><td><b>Surface End Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:2]</td></tr></table>This field specifies the ending DWord address of the buffer in Graphics Memory.</td></tr>
<tr><td></td><td>1:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
</table>

3DSTATE_SO_DECL_LIST

Source:	RenderCS
Length Bias:	2

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h GFXPIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Command SubType</b><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></td><td>26:24</td><td><b>3D Command Opcode</b><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></td><td>23:16</td><td><b>3D Command Sub Opcode</b><table><tr><td>Default Value:</td><td>17h 3DSTATE_SO_DECL_LIST</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:9</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>8:0</td><td><b>DWord Length</b><table><tr><td>Format:</td><td>=n Total Length - 2
Format: Q1</td></tr></table><table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>3h</td><td>Excludes DWord (0,1) <b>[Default]</b></td><td>Default value = 2(N-1)+3 h</td></tr></table></td></tr>
<tr><td>1</td><td>31:16</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>15:12</td><td><b>Stream to Buffer Selects [3]</b><table><tr><td>Format:</td><td>U4 bitmask
Index of SO Stream</td></tr></table>Identifies to which SO Buffers stream 3 outputs. See Stream To Buffer Selects [0] field description.</td></tr>
<tr><td></td><td>11:8</td><td><b>Stream to Buffer Selects [2]</b><table><tr><td>Format:</td><td>U4 bitmask</td></tr></table>Identifies to which SO Buffers stream 2 outputs. See Stream To Buffer Selects [0] field description.</td></tr>
<tr><td></td><td>7:4</td><td><b>Stream to Buffer Selects [1]</b><table><tr><td>Format:</td><td>U4 bitmask</td></tr></table>Identifies to which SO Buffers stream 1 outputs. See Stream To Buffer Selects [0] field description.</td></tr>
</table>

3DSTATE_SO_DECL_LIST

DWord	Bit	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

DWord	Bit	Description
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

DWord	Bit	Description
	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

DWord	Bit	Description
	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

DWord	Bit	Description
	7:0	**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).

Value	Name

3DSTATE_SO_DECL_LIST

		[0,128] entries
3..n	63:48	**SO_DECL[3,n]** Format: SO_DECL This field contains Stream 3 SO_DECL [n]
	47:32	**SO_DECL[2,n]** Format: SO_DECL This field contains Stream 2 SO_DECL [n]
	31:16	**SO_DECL[1,n]** Format: SO_DECL This field contains Stream 1 SO_DECL [n]
	15:0	**SO_DECL[0,n]** Format: SO_DECL This field contains Stream 0 SO_DECL [n]

3DSTATE_STENCIL_BUFFER

Source:	RenderCS
Length Bias:	2

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).

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).
3DSTATE_STENCIL_BUFFER must always be programmed in the along with the other Depth/Stencil state commands(i.e. 3DSTATE_DEPTH_BUFFER, 3DSTATE_CLEAR_PARAMS, or 3DSTATE_HIER_DEPTH_BUFFER)
The stencil buffer is always Tile-Y

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 Value: 1h — Name: Excludes Dword (0,1) **[Default]**
1	31	**Reserved** Format: MBZ
	30:29	**Reserved**

3DSTATE_STENCIL_BUFFER

Format:	MBZ

28:25 **Stencil Buffer Object Control State**

Format:	MEMORY_OBJECT_CONTROL_STATE

Description
Specifies the memory object control state for the stencil buffer. Stencil Buffer Object Control State [3:0]
This field is not context save and restored by hardware. If this field is programmed to any value other than zero, it must be programmed after the following commands or events: • MI_SET_CONTEXT • MI_WAIT_FOR_EVENT (Specifically waits on vblank or display flip) • Render engine goes IDLE due to head point equal to tail pointer

24:22 **Reserved**

Format:	MBZ

21:17 **Reserved**

Format:	MBZ

16:0 **Surface Pitch**

Format:	U17-1 Pitch in Bytes

This field specifies the pitch of the stencil buffer in (#Bytes - 1).

Value	Name	Description
[127, 3FFFFh]		corresponding to [128B, 128KB]also restricted to a multiple of 128B

Programming Notes
Since this surface is tiled, the pitch specified must be a multiple of the tile pitch, in the range [128B, 128KB].
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).

2 31:0 **Surface Base Address**

Format:	GraphicsAddress[31:0]Stencil_Buffer

This field specifies the starting Dword address of the buffer in mapped Graphics Memory.

Programming Notes
The Stencil Buffer can only be mapped to Main Memory (uncached).

3DSTATE_STREAMOUT

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** Format: MBZ
	7:0	**DWord Length** Default Value: 1h Format: =n Total Length - 2
1	31	**SO Function Enable** 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.
	30	**Rendering Disable** Format: U1 If set, the SO stage will not forward any topologies down the pipeline. If clear, the SO stage will forward topologies associated with Render Stream Select down the pipeline. This bit is used even if SO Function Enable is DISABLED.
	29	**Reserved** Format: MBZ
	28:27	**Render Stream Select**

3DSTATE_STREAMOUT

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.
This bit is used even if **SO Function Enable** is DISABLED.

26 **Reorder Mode**

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**

Format:	Enable

This bit controls whether StreamOutput statistics register(s) can be incremented.

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

24:23 **Reserved**

Format:	MBZ

22:12 **Reserved**

Format:	MBZ

11 **SO Buffer Enable [3]**

Format:	U1

(See SO Buffer Enable [0])

10 **SO Buffer Enable [2]**

Format:	U1

(See SO Buffer Enable [0])

9 **SO Buffer Enable [1]**

3DSTATE_STREAMOUT

		Format: U1 (See SO Buffer Enable [0])
	8	**SO Buffer Enable [0]** Format: U1 If set, stream output to SO Buffer 0 is enabled. If clear, SO Buffer 0 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. This bit is ignored if SO Function Enable is DISABLED.
	7:0	**Reserved** Format: MBZ
2	31:30	**Reserved** Format: MBZ
	29	**Stream 3 Vertex Read Offset** 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** Format: U5-1 count of 256-bit units (See Stream 0 Vertex Read Length)
	23:22	**Reserved** Format: MBZ
	21	**Stream 2 Vertex Read Offset** 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** Format: U5-1 count of 256-bit units
	15:14	**Reserved** Format: MBZ
	13	**Stream 1 Vertex Read Offset** 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**

3DSTATE_STREAMOUT		
		Format: | U5-1 count of 256-bit units (See Stream 0 Vertex Read Length)
	7:6	**Reserved** Format: | MBZ
	5	**Stream 0 Vertex Read Offset** 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** Format: | U5-1 count of 256-bit units 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).

3DSTATE_TE

Source:	RenderCS
Length Bias:	2

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** Format: MBZ
	7:0	**DWord Length** Default Value: 2h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:19	**Reserved** Format: MBZ
	18:16	**Reserved** Format: MBZ
	15:14	**Reserved** Format: MBZ
	13:12	**Partitioning** Format: U2 This field specifies how edges are partitioned based on tessellation factor. Value | Name | Description 0h | INTEGER | Outside/inside edges are divided into an integer number of equal-sized segments. 1h | ODD_FRACTIONAL | Outside/inside edges are divided into an odd number of possibly-unequal-sized segments. 2h | EVEN_FRACTIONAL | Outside/inside edges are divided into an even number of possibly-unequal-sized segments.
	11:10	**Reserved**

3DSTATE_TE

Format:	MBZ

9:8 **Output Topology**

Format:	U2

This field specifies which primitive types are to be output.

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

7:6 **Reserved**

Format:	MBZ

5:4 **TE Domain**

Format:	U2

This field specifies which type of domain is to be tessellated.

Value	Name	Description
0h	QUAD	2D (U,V) domain is tessellated
1h	TRI	Triangular (U,V,W) domain is tessellated
2h	ISOLINE	2D (U,V) domain is tessellated.

3 **Reserved**

Format:	MBZ

2:1 **TE Mode**

Format:	U2

When TE Enable is ENABLED, this field specifies the overall operation of the TE stage. This field is ignored if TE Enable is DISABLED.

Value	Name	Description
0h	HW_TESS	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.
1h	SW_TESS	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.

3DSTATE_TE

2h	Reserved	Reserved
3h	Reserved	Reserved

0

TE Enable

Format:	Enable

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.

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.

2 31:0

Maximum Tessellation Factor Odd

Format:	IEEE_Float

This field specifies the maximum TessFactor for ODD_FRACTIONAL partitioning when in HW_TESS mode.

Value	Name	Description
427c0000h	63	Per API Spec, For normal operation software should set this value to 63.0
[40400000h,427c0000h]	Reserved	Reserved.

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

3 31:0

Maximum Tessellation Factor Not Odd

Format:	IEEE_Float

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

Value	Name	Description
42800000h	64	Per API Spec, For normal operation software should set this value to 64.0
[40000000h,42800000h]	Reserved	Reserved

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

3DSTATE_URB_DS

Source:	RenderCS
Length Bias:	2

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.

Programming Notes
3DSTATE_URB_VS, 3DSTATE_URB_HS, and 3DSTATE_URB_GS must also be programmed in order for the programming of this state to be valid.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 32h 3DSTATE_URB_DS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31	**Reserved** Format: MBZ
	30	**Reserved** Format: MBZ
	29:25	**DS URB Starting Address** Format: U5 Offset from the start of the URB memory where DS starts its allocation, specified in multiples of 8 KB. Value | Name [0,11] |
	24:16	**DS URB Entry Allocation Size** Format: U9-1 Count of 512-bit units Specifies the length of each URB entry owned by DS. This field is always used (even if DS

3DSTATE_URB_DS

Function Enable is DISABLED).

Value	Name
[0,9]	

15:0 **DS Number of URB Entries**

Description
Specifies the number of URB entries that are used by DS. 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 10 URB entries.

Value	Name
[0,288]	

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

3DSTATE_URB_GS

Source:	RenderCS
Length Bias:	2

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.

Programming Notes
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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h GFXPIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Command SubType</b><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></td><td>26:24</td><td><b>3D Command Opcode</b><table><tr><td>Default Value:</td><td>0h 3DSTATE_PIPELINED</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>23:16</td><td><b>3D Command Sub Opcode</b><table><tr><td>Default Value:</td><td>33h 3DSTATE_URB_GS</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>0h DWORD_COUNT_n</td></tr><tr><td>Format:</td><td>=n</td></tr></table></td></tr>
<tr><td>1</td><td>31</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>30</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>29:25</td><td><b>GS URB Starting Address</b><table><tr><td>Format:</td><td>U5</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></tr><tr><td>[0,11]</td><td></td></tr></table></td></tr>
<tr><td></td><td>24:16</td><td><b>GS URB Entry Allocation Size</b><table><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</td></tr>
</table>

<table>
<tr><th colspan="3">3DSTATE_URB_GS</th></tr>
<tr><td></td><td></td><td>Function Enable is DISABLED).</td></tr>
<tr><td></td><td>15:0</td><td><b>GS Number of URB Entries</b>
Specifies the number of URB entries that are used by GS. This field is always used (even if GS Function Enable is DISABLED).
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>[0,192]</td><td></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

Source:	RenderCS
Length Bias:	2

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.

Programming Notes
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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h GFXPIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Command SubType</b><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></td><td>26:24</td><td><b>3D Command Opcode</b><table><tr><td>Default Value:</td><td>0h 3DSTATE_PIPELINED</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>23:16</td><td><b>3D Command Sub Opcode</b><table><tr><td>Default Value:</td><td>31h 3DSTATE_URB_HS</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>0h DWORD_COUNT_n</td></tr><tr><td>Format:</td><td>=n</td></tr></table></td></tr>
<tr><td>1</td><td>31</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>30</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>29:25</td><td><b>HS URB Starting Address</b><table><tr><td>Format:</td><td>U5</td></tr></table>Offset from the start of the URB memory where HS starts its allocation, specified in multiples of 8 KB.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,11]</td><td></td></tr></table></td></tr>
<tr><td></td><td>24:16</td><td><b>HS URB Entry Allocation Size</b><table><tr><td>Format:</td><td>U9-1 Count of 512-bit units</td></tr></table>Specifies the length of each URB entry owned by HS. This field is always used (even if HS</td></tr>
</table>

<table>
<tr><th colspan="3">3DSTATE_URB_HS</th></tr>
<tr><td></td><td></td><td>Function Enable is DISABLED).</td></tr>
<tr><td></td><td>15:0</td><td><b>HS Number of URB Entries</b>
Specifies the number of URB entries that are used by HS. 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"
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>[0,32]</td><td></td></tr>
</table>
</td></tr>
</table>

3DSTATE_URB_VS

Source:	RenderCS
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.

Programming Notes
3DSTATE_URB_HS, 3DSTATE_URB_DS, and 3DSTATE_URB_GS must also be programmed in order for the programming of this state to be valid.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 30h 3DSTATE_URB_VS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31	**Reserved** Format: MBZ
	30	**Reserved** Format: MBZ
	29:25	**VS URB Starting Address** Format: U5 Offset from the start of the URB memory where VS starts its allocation, specified in multiples of 8 KB. **Value** | **Name**

<table>
<tr><th colspan="3">3DSTATE_URB_VS</th></tr>
<tr><td></td><td></td><td><table><tr><td>[0,11]</td><td></td></tr></table></td></tr>
<tr><td></td><td>24:16</td><td><b>VS URB Entry Allocation Size</b>
<table><tr><td>Format:</td><td>U9-1 count of 512-bit units</td></tr></table>
Specifies the length of each URB entry owned by VS. This field is always used (even if VS Function Enable is DISABLED).
<table><tr><th>Programming Notes</th></tr><tr><td>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.</td></tr></table></td></tr>
<tr><td></td><td>15:0</td><td><b>VS Number of URB Entries</b>
<table><tr><td>Format:</td><td>U16</td></tr></table>
Specifies the number of URB entries that are used by VS. This field is always used (even if VS Function Enable is DISABLED).
<table><tr><th>Value</th><th>Name</th></tr><tr><td>[32,512]</td><td></td></tr></table>
<table><tr><th>Programming Notes</th></tr><tr><td>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"</td></tr></table></td></tr>
</table>

3DSTATE_VERTEX_BUFFERS

Source:	RenderCS
Length Bias:	2

Description
This command is used to specify VB state used by the VF function.
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
	15:8	**Reserved**
	7:0	**DWord Count** Default Value: 3 DWORD_COUNT_n Format: =n n = 4b-1 (where b = # of buffer states included)

3DSTATE_VERTEX_BUFFERS

1..n	127:0	**Vertex Buffer State [n]** Format: VERTEX_BUFFER_STATE

3DSTATE_VERTEX_ELEMENTS

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.
Up to 34 elements.

Programming Notes
At least one VERTEX_ELEMENT_STATE structure must be included.
Inclusion of partial VERTEX_ELEMENT_STATE structures is UNDEFINED.
SW must ensure that at least one vertex element is defined prior to issuing a 3DPRIMTIVE command, or operation is UNDEFINED.
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'.
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.
See additional restrictions listed in the command fields and VERTEX_ELEMENT_STATE description.
Element[0] must be valid.
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).
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
	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

3DSTATE_VERTEX_ELEMENTS

<table>
<tr><td></td><td></td><td><table><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td></td><td>15:8</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Count</b><table><tr><td>Format:</td><td>=n</td></tr></table>Vertex Element Count = (DWord Count + 1) / 2<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>1</td><td>DWORD_COUNT_n <b>[Default]</b></td><td>excludes DWords 0,1</td></tr><tr><td>[1,66]</td><td>Range</td><td>1-34 Elements</td></tr></table></td></tr>
<tr><td>1..n</td><td>63:0</td><td><b>Element [n]</b><table><tr><td>Format:</td><td>VERTEX_ELEMENT_STATE</td></tr></table></td></tr>
</table>

3DSTATE_VF_STATISTICS

Source:	RenderCS
Length Bias:	1

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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h GFXPIPE</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Command SubType</b><table><tr><td>Format:</td><td>Opcode</td></tr></table><table><tr><th>Value</th><th>Name</th></tr><tr><td>1h</td><td>Pipelined, Single DWord <b>[Default]</b></td></tr></table></td></tr>
<tr><td></td><td>26:24</td><td><b>3D Command Opcode</b><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></td><td>23:16</td><td><b>3D Command Sub Opcode</b><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></td><td>15:1</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>0</td><td><b>Statistics Enable</b><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_VIEWPORT_STATE_POINTERS_CC

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** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:5	**CC Viewport Pointer** 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** Format: MBZ

3DSTATE_VIEWPORT_STATE_POINTERS_SF_CLIP

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** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:6	**SF Clip Viewport Pointer** 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** Format: MBZ

3DSTATE_VS

Source:	RenderCS
Length Bias:	2

Description
The state used by VS is defined with this inline state packet.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h GFXPIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Command SubType</b><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></td><td>26:24</td><td><b>3D Command Opcode</b><table><tr><td>Default Value:</td><td>0h 3DSTATE_PIPELINED</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>23:16</td><td><b>3D Command Sub Opcode</b><table><tr><td>Default Value:</td><td>10h 3DSTATE_VS</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>4h Excludes DWord (0,1)</td></tr><tr><td>Format:</td><td>=n Total Length - 2</td></tr></table></td></tr>
<tr><td>1</td><td>31:6</td><td><b>Kernel Start Pointer</b><table><tr><td>Format:</td><td>InstructionBaseOffset[31:6]Kernel</td></tr></table>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. This field is ignored if VS Function Enable is DISABLED.</td></tr>
<tr><td></td><td>5:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>2</td><td>31</td><td><b>Single Vertex Dispatch</b><table><tr><td>Format:</td><td>U1 Enumerated type</td></tr></table>This field can be used to force single vertex SIMD4x2 VS threads.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Multiple</td><td>Dual vertex SIMD4x2 thread dispatches are allowed.</td></tr><tr><td>1h</td><td>Single</td><td>Single vertex SIMD4x2 thread dispatches are forced.</td></tr></table></td></tr>
<tr><td></td><td>30</td><td><b>Vector Mask Enable (VME)</b>
When SPF=0, 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.</td></tr>
</table>

3DSTATE_VS

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

29:27 **Sampler Count**

Specifies how many samplers (in multiples of 4) the vertex shader 0 kernel uses. Used only for prefetching the associated sampler state entries. This field is ignored if VS Function Enable is DISABLED.

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

26 **Reserved**

Format:	MBZ

25:18 **Binding Table Entry Count**

Format:	U8

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.

Value	Name
[0,255]	

17 **Reserved**

Format:	MBZ

16 **Floating Point Mode**

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
0h	IEEE-754	Use IEEE-754 Rules
1h	Alternate	Use alternate rules

15:14 **Reserved**

Format:	MBZ

13 **Illegal Opcode Exception Enable**

Format:	Enable

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

3DSTATE_VS

	12	**Reserved** Format: MBZ
	11:8	**Reserved** Format: MBZ
	7	**Software Exception Enable** Format: Enable This bit gets loaded into EU CR0.1[13] (note the bit # difference). See Exceptions and ISA Execution Environment.This field is ignored if VS Function Enable is DISABLED.
	6:0	**Reserved** Format: MBZ
3	31:10	**Scratch Space Base Offset** Format: GeneralStateOffset[31:10]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. This field is ignored if VS Function Enable is DISABLED. **Value** | **Name** | **Description** [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.
4	31:25	**Reserved** Format: MBZ
	24:20	**Dispatch GRF Start Register for URB Data** Format: U5

3DSTATE_VS

Specifies the starting GRF register number for the URB portion (Constant + Vertices) of the thread payload. This field is ignored if VS Function Enable is DISABLED.

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

19:17 **Reserved**

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.

Value	Name
[1,63]	

Programming Notes
It is UNDEFINED to set this field to 0 indicating no Vertex URB data to be read and passed to the thread.

10 **Reserved**

Format:	MBZ

9:4 **Vertex URB Entry Read Offset**

Format:	U6

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

Value	Name
[0,63]	

3:0 **Reserved**

Format:	MBZ

5

31:25 **Maximum Number of Threads**

Format:	U7-1 representing 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
[0,15]	indicating thread count of [1,16]

24:23 **Reserved**

3DSTATE_VS

		Format: MBZ
	22:11	**Reserved** Format: MBZ
	10	**Statistics Enable** Format: Enable **Description** If ENABLED, this FF unit will engage in statistics gathering. See the Statistics Gathering section later in this chapter. If DISABLED, statistics information associated with this FF stage will be left unchanged. This field is used even if VS Function Enable is DISABLED.
	9:3	**Reserved** Format: MBZ
	2	**Reserved** Format: MBZ
	1	**Vertex Cache Disable** 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, 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, though pass thru the VS stage unmodified (not shaded). 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.
	0	**VS Function Enable** Format: Enable **Description** 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 sent down the pipeline unmodified. The Vertex Cache is still available in this mode, if enabled. If Statistics Enable is ENABLED, VS_INVOCATION_COUNT will increment by 1 for every vertex that passes through the VS stage, even if VS Function Enable is DISABLED. This field is always used.

3DSTATE_WM

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** Format: MBZ
	7:0	**DWord Length** Default Value: 01h Excludes DWord (0,1) Format: =n Total Length - 2
1	31	**Statistics Enable** 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.
	30	**Depth Buffer Clear** Format: Enable When set, the depth buffer is initialized as a side-effect of rendering pixels. **Programming Notes** If this field is enabled, 2. **the Depth Test Enable** field in DEPTH_STENCIL_STATE must be disabled. 3. 3DSTATE_DEPTH_BUFFER::Depth Write Enable must be set. 4. 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.

3DSTATE_VS

		2. DEPTH_STENCIL_STATE::Stencil Write Mask must be 0xFF 3. DEPTH_STENCIL_STATE::Stencil Test Mask must be 0xFF 4. DEPTH_STENCIL_STATE::Back Face Stencil Write Mask must be 0xFF 5. 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	**Thread Dispatch Enable** Format: | Enable This bit, if set, indicates that it is possible for a PS thread to modify a render target, i.e.,at least one render target is enabled (is not of type SURFTYPE_NULL and has at least one channel enabled for writes) and the PS kernel contains a code path that may issue a write to that/those enabled RTs. **Programming Notes** This bit is used for performance optimizations and does not directly control writing to render targets. If this bit is DISABLED, no pixel shader threads will be dispatched. For correct behavior, this bit must be set consistently with the behavior of the PS kernel, i.e. if this bit is DISABLED the PS kernel must not write color or depth to any render targets. If this field is disabled, **Pixel Shader Kill Pixel** must be disabled.
	28	**Depth Buffer Resolve Enable** 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, 2. the **Depth Buffer Clear** and **Hierarchical Depth Buffer Resolve Enable** fields must both be disabled. 3. 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	**Hierarchical Depth Buffer Resolve Enable** 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,

<table>
<tr><th colspan="3">3DSTATE_VS</th></tr>
<tr><td></td><td></td><td><table><tr><td>2. the <b>Depth Buffer Clear</b> and <b>Depth Buffer Resolve Enable</b> fields must both be disabled.
3. 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 <b>Hierarchical Depth Buffer Enable</b> is disabled, enabling this field will have no effect.
<b>Performance Note:</b> 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>Software needs to do an ambiguate after allocating the surface for the first time if the depth buffer width and height are NOT aligned to 8 and 4 respectively.</td></tr></table></td></tr>
<tr><td></td><td>26</td><td><b>Legacy Diamond Line Rasterization</b><table><tr><td>Format:</td><td>Enable</td></tr></table>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).</td></tr>
<tr><td></td><td>25</td><td><b>Pixel Shader Kill Pixel</b><table><tr><td>Format:</td><td>Enable</td></tr></table>This bit, if ENABLED, indicates that the PS kernel or color calculator 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.
• A sampler with chroma key enabled with kill pixel mode is used by the pixel shader.
• Any render target has <b>Alpha Test Enable</b> or <b>AlphaToCoverage Enable</b> enabled.
• The pixel shader kernel generates and outputs oMask.

<b>Note:</b> As ClipDistance clipping is fully supported in hardware and therefore not via PS instructions, there should be no need to ENABLE this bit <u>due to ClipDistance clipping</u>.</td></tr>
<tr><td></td><td>24:23</td><td><b>Pixel Shader Computed Depth Mode</b><table><tr><td>Format:</td><td>U2 Enumerated Type</td></tr></table>This field specifies the computed depth mode for the pixel shader.</td></tr>
</table>

3DSTATE_VS

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
When bit 5 is set in WM_STATE(i.e. RT independent rasterization is enabled), this field can not be programmed to values: 2h or 3h.

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

Format:	U2 Enumerated Type

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

Value	Name	Description
0h	EDSC_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)
1h	EDSC_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)
2h	EDSC_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.
3h	Reserved	

Programming Notes
If EDSC_PSEXEC mode is selected, **Thread Dispatch Enable** must be set.

Restriction
Restriction: When value of "2h" is programmed, PS_INVOCATIONs_COUNT may not be accurate.

20 **Pixel Shader Uses Source Depth**

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.

19 **Pixel Shader Uses Source W**

3DSTATE_VS

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.

18:17 **Position ZW Interpolation Mode**

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

Programming Notes
When bit 5 is set in WM_STATE, value of 3h is not defined for this field. Programming Note: When bit 5 in dword 1 (RT Independent Rasterization Enable) is set and bit 30 in dword 2 (PS UAV-only) is not set in WM_STATE, value of 3h is not defined for this field.

16:11 **Barycentric Interpolation Mode**

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 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. Restriction: When Centroid Barycentric mode is required, HW may produce incorrect interpolation results when a 2X2 pixels have unlit pixels.

10 **Pixel Shader Uses Input Coverage Mask**

Format:	Enable

This bit, if ENABLED, indicates that the PS kernel requires the input coverage mask to be passed in the payload.

3DSTATE_VS

9:8 **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
1h		1.0 pixels
2h		2.0 pixels
3h		4.0 pixels

7:6 **Line Antialiasing Region Width**

Format:	U2

This field specifies the distance over which the anti-aliased line coverage is computed.

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

5 **Reserved**

Format:	MBZ

4 **Polygon Stipple Enable**

Format:	Enable

Enables the Polygon Stipple function.

3 **Line Stipple Enable**

Format:	Enable

Enables the Line Stipple function.

2 **Point Rasterization Rule**

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
0h	RASTRULE_UPPER_LEFT	To match "normal" upper left rules for surface primitives
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).

1:0 **Multisample Rasterization Mode**

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, the API's multisample enable

<table>
<tr><th colspan="3">3DSTATE_VS</th></tr>
<tr><td></td><td></td><td>state setting (if any), and whether 1X or 4X MSRTs are bound. This state is duplicated in 3DSTATE_SF and both must be set to the same value. Refer to the "Multisampling" section for details on the settings of this field.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0h</td><td>MSRASTMODE_OFF_PIXEL</td></tr>
<tr><td>1h</td><td>MSRASTMODE_OFF_PATTERN</td></tr>
<tr><td>2h</td><td>MSRASTMODE_ON_PIXEL</td></tr>
<tr><td>3h</td><td>MSRASTMODE_ON_PATTERN</td></tr>
</table></td></tr>
<tr><td>2</td><td>31</td><td><b>Multisample Dispatch Mode</b>
<table>
<tr><td>Format:</td><td>U1 Enumerated Type</td></tr>
</table>
This bit, along with <b>Number of Multisamples</b>, determines how PS threads are dispatched. Software programs this bit depending on the per-pixel v.s per-sample PS execution requirement. When <b>RT Independent Rasterization Enable = 1, value of 0h for this field is not allowed.</b>
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>MSDISPMODE_PERSAMPLE</td><td>This is the high-quality DX10.1 multisample mode where (over and above PERPIXEL mode) the PS is run for each covered sample. This mode is also used for "normal" non-multisample rendering (aka 1X), given Number of Multisamples is programmed to NUMSAMPLES_1.</td></tr>
<tr><td>1h</td><td>MSDISPMODE_PERPIXEL</td><td>This is the classic multisample mode of operation, typically used for both antialiasing and transparency. Setup and rasterization operate in full multisample mode, testing coverage and depth/stencil test at the sample level but only running the PS once per pixel.</td></tr>
</table></td></tr>
<tr><td></td><td>30:0</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
</table>

add - Addition

Source:	EuIsa
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
*B,*W,*D	*B,*W,*D
*B,*W,*D	F
F	F
DF	DF

DWord	Bit	Description
0..3	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM
	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

addc - Addition with Carry

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

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	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM
	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

asr - Arithmetic Shift Right

Source:	EuIsa
Length Bias:	4

Description
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.
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.
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 = src1.chan[n] & 0x1F; // Always use low 5 bits for shift count.
        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
*B,*W,*D	*B,*W,*D

DWord	Bit	Description
0..3	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM
	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

avg - Average

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	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM
	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

bfe - Bit Field Extract

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
Restriction: No accumulator access, implicit or explicit.
Restriction: All three-source instructions have certain restrictions, described in Instruction Machine Formats.

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
	105	**Reserved** Format: MBZ

<table>
<tr><th colspan="3">bfe - Bit Field Extract</th></tr>
<tr><td></td><td>104:85</td><td><b>Source 1</b><table><tr><td>Format:</td><td>EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC</td></tr></table></td></tr>
<tr><td></td><td>84</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>83:64</td><td><b>Source 0</b><table><tr><td>Format:</td><td>EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC</td></tr></table></td></tr>
<tr><td></td><td>63:56</td><td><b>Destination Register Number</b><table><tr><td>Format:</td><td>DstRegNum</td></tr></table></td></tr>
<tr><td></td><td>55:53</td><td><b>Destination Subregister Number</b><table><tr><td>Format:</td><td>DstSubRegNum[2:0]</td></tr></table></td></tr>
<tr><td></td><td>52:49</td><td><b>Destination Channel Enable</b><table><tr><td>Format:</td><td>ChanEn[4]</td></tr></table>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 <i>x</i>, <i>y</i>, <i>z</i>, and <i>w</i>, respectively, where <i>x</i> corresponds to Channel 0 in the group and <i>w</i> corresponds to channel 3 in the group</td></tr>
<tr><td></td><td>48</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>47</td><td><b>NibCtrl</b><table><tr><td>Format:</td><td>NibCtrl</td></tr></table></td></tr>
<tr><td></td><td>46</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>45:44</td><td><b>Destination Data Type</b>
This field contains the data type for the destination<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>Single Precision Float</td></tr><tr><td>01b</td><td>DWord</td></tr><tr><td>10b</td><td>Unsigned DWord</td></tr><tr><td>11b</td><td>Double Precision Float</td></tr></table></td></tr>
<tr><td></td><td>43:42</td><td><b>Source Data Type</b>
This field contains the data type for all three sources<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>Single Precision Float</td></tr><tr><td>01b</td><td>DWord</td></tr><tr><td>10b</td><td>Unsigned DWord</td></tr><tr><td>11b</td><td>Double Precision Float</td></tr></table></td></tr>
<tr><td></td><td>41:40</td><td><b>Source 2 Modifier</b></td></tr>
</table>

bfe - Bit Field Extract

		Exists If: ([Property[Source Modification]=='true') Format: SrcMod
	39:38	**Source 1 Modifier** Exists If: ([Property[Source Modification]=='true') Format: SrcMod
	41:36	**Reserved** Exists If: ([Property[Source Modification]=='false') Format: MBZ
	37:36	**Source 0 Modifier** Exists If: ([Property[Source Modification]=='true') Format: SrcMod
	35	**Reserved** Format: MBZ
	34	**Flag Register Number** This field contains the flag register number for instructions with a non-zero Conditional Modifier.
	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

bfi1 - Bit Field Insert 1

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

bfi1 - Bit Field Insert 1

0..3	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM
	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

bfi2 - Bit Field Insert 2

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
Restriction: No accumulator access, implicit or explicit.
Restriction: All three-source instructions have certain restrictions, described in Instruction Machine Formats.

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

bfi2 - Bit Field Insert 2

<table>
<tr><td>0..3</td><td>127:126</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>125:106</td><td><b>Source 2</b><table><tr><td>Format:</td><td>EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC</td></tr></table></td></tr>
<tr><td></td><td>105</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>104:85</td><td><b>Source 1</b><table><tr><td>Format:</td><td>EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC</td></tr></table></td></tr>
<tr><td></td><td>84</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>83:64</td><td><b>Source 0</b><table><tr><td>Format:</td><td>EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC</td></tr></table></td></tr>
<tr><td></td><td>63:56</td><td><b>Destination Register Number</b><table><tr><td>Format:</td><td>DstRegNum</td></tr></table></td></tr>
<tr><td></td><td>55:53</td><td><b>Destination Subregister Number</b><table><tr><td>Format:</td><td>DstSubRegNum[2:0]</td></tr></table></td></tr>
<tr><td></td><td>52:49</td><td><b>Destination Channel Enable</b><table><tr><td>Format:</td><td>ChanEn[4]</td></tr></table>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 <i>x</i>, <i>y</i>, <i>z</i>, and <i>w</i>, respectively, where <i>x</i> corresponds to Channel 0 in the group and <i>w</i> corresponds to channel 3 in the group</td></tr>
<tr><td></td><td>48</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>47</td><td><b>NibCtrl</b><table><tr><td>Format:</td><td>NibCtrl</td></tr></table></td></tr>
<tr><td></td><td>46</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>45:44</td><td><b>Destination Data Type</b>
This field contains the data type for the destination<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>Single Precision Float</td></tr><tr><td>01b</td><td>DWord</td></tr><tr><td>10b</td><td>Unsigned DWord</td></tr><tr><td>11b</td><td>Double Precision Float</td></tr></table></td></tr>
<tr><td></td><td>43:42</td><td><b>Source Data Type</b></td></tr>
</table>

bfi2 - Bit Field Insert 2

		This field contains the data type for all three sources **Value** — **Name** 00b — Single Precision Float 01b — DWord 10b — Unsigned DWord 11b — Double Precision Float
	41:40	**Source 2 Modifier** Exists If: ([Property[Source Modification]=='true') Format: SrcMod
	39:38	**Source 1 Modifier** Exists If: ([Property[Source Modification]=='true') Format: SrcMod
	41:36	**Reserved** Exists If: ([Property[Source Modification]=='false') Format: MBZ
	37:36	**Source 0 Modifier** Exists If: ([Property[Source Modification]=='true') Format: SrcMod
	35	**Reserved** Format: MBZ
	34	**Flag Register Number** This field contains the flag register number for instructions with a non-zero Conditional Modifier.
	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

bfrev - Bit Field Reverse

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
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	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32
	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

brc - Branch Converging

Source:	EuIsa
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 are at location src1 when immediates and at location src0 when reg32, where reg32 is accessed as a scalar DWord containing both JIP and UIP. The null register must be used (for example, by the assembler) as dst. When offsets are immediate, src0 must be null.
Format: [(pred)] brc (exec_size) JIP UIP

Restriction
Restriction: A brc instruction must use the Switch instruction option.

Syntax

```
[(pred)] brc (exec_size) imm16 imm16 [(pred)] brc (exec_size) reg32
```

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);
}
```

Predication	Conditional Modifier	Saturation	Source Modifier	Source Types
Y	N	N	N	D

DWord	Bit	Description
0..3	127:112	**UIP** Format: S15 The jump distance in number of eight-byte units if a jump is taken for the channel.
	111:96	**JIP** Format: S15 The jump distance in number of eight-byte units if a jump is taken for the instruction.
	95:64	**Reserved** Format: MBZ
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

brd - Branch Diverging

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 null register must be used at dst locations.
Format: [(pred)] brd (exec_size) JIP

Restriction
Restriction: A brd instruction must use the Switch instruction option.

Syntax

```
[(pred)] brd (exec_size) imm16 [(pred)] brd (exec_size) reg32
```

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); }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

Src Types
D

DWord	Bit	Description
0..3	127:112	**Reserved** Format: MBZ
	111:96	**JIP** Format: S15 Jump Target Offset. The relative offset in 64-bit units if a jump is taken for the instruction.
	95:91	**Reserved** Format: MBZ
	90	**Flag Register Number** Added a second flag register
	89	**Flag Subregister Number** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and

brd - Branch Diverging

		conditional destination, if both predication and conditional modifier are enabled.
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN1
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align16') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN16
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

break - Break

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 16-bit instruction pointer offsets. Both the JIP and UIP are signed 16-bit numbers, added to IP pre-increment. In GEN binary, JIP and UIP are at location src1 and must be of type W (signed word integer).

Format:
[(pred)] break (exec_size) JIP UIP

Syntax

```
[(pred)] break (exec_size) imm16 imm16
```

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); }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

DWord	Bit	Description
0..3	127:112	**UIP** Format: S15 The jump distance in number of eight-byte units if a jump is taken for the channel.
	111:96	**JIP** Format: S15 The jump distance in number of eight-byte units if a jump is taken for the instruction.
	95:64	**Reserved** Format: MBZ
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS

break - Break

	31:0	**Header** Format: EU_INSTRUCTION_HEADER

call - Call

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 must be a signed immediate operand. When a jump occurs, this value is added to IP pre-increment.

Format:
[(pred)] call (exec_size) dst JIP

Restriction
Restriction: The call instruction must have DWord source and destination type, and the destination must be QWord aligned.
Restriction: The source0 regioning control must be < 2;2,1 > .
Restriction: The execution size must be 2.

Syntax

```
[(pred)] call (exec_size) reg imm16
```

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);
}
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

Dst Types
D,UD

DWord	Bit	Description
0..3	127:112	**Reserved**

DWord	Bit	Description
		Format: MBZ
	111:96	**JIP** Format: S15 Jump Target Offset. The relative offset in 64-bit units if a jump is taken for the instruction.
	95:91	**Reserved** Format: MBZ
	90	**Flag Register Number** Added a second flag register
	89	**Flag Subregister Number** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN1
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align16') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN16
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

COLOR_BLT

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 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: 40h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. Value — Name 1xb — Write Alpha Channel x1b — Write RGB Channel
	19:6	**Reserved** Format: MBZ
	5:0	**DWord Length** Default Value: 03h
1 BR13	31:26	**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 (Signed)** Destination pitch in bytes (Same as before).

COLOR_BLT

2 BR14	31:16	**Destination Height (in scan lines)**
	15:0	**Destination Byte Width (in bytes)**
3 BR09	31:0	**Destination Address** Format: GraphicsAddress[31:0] Address of the first byte to be written.
4 BR16	31:0	**Solid Pattern Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]

cmp - Compare

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
Restriction: Accumulator cannot be destination, implicit or explicit. The destination must be a general register or the null register.
Restriction: A SIMD16 instruction is not allowed for DWord data types. Use two SIMD8 instructions.
Restriction: If the destination is the null register, the {Switch} instruction option must be used.

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
*B,*W,*D	*B,*W,*D
*B,*W,*D	F
F	F
DF	DF

DWord	Bit	Description
0..3	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM
	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

cmpn - Compare NaN

Source:	EuIsa
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
Restriction: Accumulator cannot be destination, implicit or explicit. The destination must be a general register or the null register.
Restriction: A SIMD16 instruction is not allowed for DWord data types. Use two SIMD8 instructions.
Restriction: If the destination is the null register, the {Switch} instruction option must be used.

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
*B,*W,*D	*B,*W,*D
*B,*W,*D	F
F	F
DF	DF

DWord	Bit	Description

cmpn - Compare NaN

0..3	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM
	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

sendc - Conditional Send Message

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
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
	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
	88:64	**Source 0**		
			Exists If:	(Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align16')
			Format:	EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN16
	63:32	**Operand Control**		
			Format:	EU_INSTRUCTION_OPERAND_CONTROLS
	31:28	**Controls B**		
			Format:	EU_INSTRUCTION_CONTROLS_B
	27:24	**Shared Function ID (SFID)**		
			Format:	SFID
	23:8	**Controls A**		
			Format:	EU_INSTRUCTION_CONTROLS_A

sendc - Conditional Send Message			
	7	**Reserved**	
		Format:	MBZ
	6:0	**Opcode**	
		Format:	EU_OPCODE

cont - Continue

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 16-bit instruction pointer offsets. Both the JIP and UIP are signed 16-bit numbers, added to IP pre-increment. In GEN binary, JIP and UIP are at location src1 and must be of type W (signed word integer).

Format:
[(pred)] cont (exec_size) JIP UIP

Restriction

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) imm16 imm16
```

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);
}
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

DWord	Bit	Description
0..3	127:112	**UIP** Format: S15 The jump distance in number of eight-byte units if a jump is taken for the channel.
	111:96	**JIP** Format: S15 The jump distance in number of eight-byte units if a jump is taken for the instruction.

cont - Continue

	95:64	**Reserved** Format: MBZ
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

cbit - Count Bits Set

Source:	EuIsa
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

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	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32
	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

dp2 - Dot Product 2

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
Restriction: Execution size cannot be less than 4.
Restriction: Horizontal strides must be 1.
Restriction: 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
Y	Y	Y	Y

Src Types	Dst Types
F	F

DWord	Bit	Description
0..3	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM
	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS

dp2 - Dot Product 2

	31:0	**Header** Format: EU_INSTRUCTION_HEADER

dp3 - Dot Product 3

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
Restriction: Execution size cannot be less than 4.
Restriction: Horizontal strides must be 1.
Restriction: 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	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM
	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS

dp3 - Dot Product 3

	31:0	**Header** Format: EU_INSTRUCTION_HEADER

dp4 - Dot Product 4

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
Restriction: Execution size cannot be less than 4.
Restriction: Horizontal strides must be 1.
Restriction: 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	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM
	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

dph - Dot Product Homogeneous

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
Restriction: Execution size cannot be less than 4.
Restriction: Horizontal strides must be 1.
Restriction: 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	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM
	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

else - Else

Source:	EuIsa
Length Bias:	4

The else instruction is an optional statement within an if/else/endif block of code. It restricts execution within the else/endif portion to the opposite set of channels enabled under the if/else portion. Channels which were inactive before entering the if/endif block remain inactive throughout the entire block.

All enabled channels upon arriving at the else instruction are redirected to the matching endif. If all channels are redirected (by else or before else), a relative jump is performed to the location specified by JIP. The jump target should be the the matching endif instruction for that conditional block.

The following table describes the 16-bit JIP. In GEN binary, JIP is at location src1 and must be of type W (signed word integer). JIP must be an immediate operand, it is a signed 16-bit number and is intended to be forward referencing. This value is added to IP pre-increment.

Format:
else (exec_size) JIP

Restriction
Restriction: Predication is not allowed.
Restriction: The execution size must be the same for the if, else, and endif instructions of the same code block.

Syntax
`else (exec_size) imm16`

Pseudocode

```
Evaluate(WrEn);
for ( n = 0; n < 32; n++ ) {
    if ( WrEn.channel[n] ) {
        PcIP[n] = IP + JIP;
    }
}
if ( PcIP != (IP + 1) ) { // for all channels
    Jump(IP + JIP);
}
```

Predication	Conditional Modifier	Saturation	Source Modifier
N	N	N	N

DWord	Bit	Description
0..3	127:112	**UIP** Format: S15 The jump distance in number of eight-byte units if a jump is taken for the channel.
	111:96	**JIP** Format: S15 The jump distance in number of eight-byte units if a jump is taken for the instruction.
	95:64	**Reserved** Format: MBZ
	63:32	**Operand Control**

else - Else

		Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

endif - End If

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 16-bit JIP. In GEN binary, JIP is at location src1 and must be of type W (signed word integer). JIP must be an immediate operand, it is a signed 16-bit number. This value is added to IP pre-increment.
Format: endif JIP

Restriction
Restriction: Predication is not allowed.
Restriction: The execution size must be the same for the if, else, and endif instructions of the same code block.

Syntax
`endif (exec_size) imm16`

Pseudocode

```
Evaluate(WrEn); if ( WrEn == 0 ) { // all channels false Jump(IP + JIP); }
```

Predication	Conditional Modifier	Saturation	Source Modifier
N	N	N	N

DWord	Bit	Description
0..3	127:112	**Reserved** Format: MBZ
	111:96	**JIP** Format: S15 Jump Target Offset. The relative offset in 64-bit units if a jump is taken for the instruction.
	95:91	**Reserved** Format: MBZ
	90	**Flag Register Number** Added a second flag register
	89	**Flag Subregister Number** This field specifies the sub-register number for a flag register operand. There are two sub-

endif - End If

		registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN1
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align16') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN16
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

math - Extended Math Function

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
Restriction: Accumulator access is allowed only for ieee macro functions.
Restriction: The math instruction does not support indirect addressing modes.
Restriction: The only supported rounding mode for math instruction is Round to Nearest Even.
Restriction: INT DIV function does not support SIMD16.
Restriction: The FDIV function is not supported in ALT_MODE.
Restriction: The math instruction must use GRF registers as source(s) and destination.
Restriction: The supported regioning mode for math instruction is align1 with the following restrictions: Scalar source is supported. Source and destination horizontal stride must be 1. Width must be the same as execution size. Source and destination offset must be the same, except the case of scalar source.

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
```

math - Extended Math Function

```
                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;
            }
       }
    }
}
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	Y	N

Src Types	Dst Types
F	F
D	D
UD	UD

DWord	Bit	Description
0..3	127:64	**RegSource** Format: EU_INSTRUCTION_SOURCES_REG_REG
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:28	**Controls B** Format: EU_INSTRUCTION_CONTROLS_B
	27:24	**Function Control (FC)** Format: FC
	23:8	**Controls A** Format: EU_INSTRUCTION_CONTROLS_A
	7	**Reserved** Format: MBZ
	6:0	**Opcode** Format: EU_OPCODE

fbl - Find First Bit from LSB Side

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

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	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32
	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

fbh - Find First Bit from MSB Side

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
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	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32
	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM')

fbh - Find First Bit from MSB Side

		Format: EU_INSTRUCTION_SOURCES_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

frc - Fraction

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

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	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32
	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

GPGPU_OBJECT

Source:	RenderCS
Length Bias:	2

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h GFXPIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Pipeline</b><table><tr><td>Default Value:</td><td>2h Media</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>26:24</td><td><b>Media Command Opcode</b><table><tr><td>Default Value:</td><td>1h GPGPU_OBJECT</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>23:16</td><td><b>SubOpcode</b><table><tr><td>Default Value:</td><td>04h GPGPU_OBJECT SubOp</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:9</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>8</td><td><b>Predicate Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>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><b>DWord Length</b><table><tr><td>Format:</td><td>=n Total Length -2</td></tr></table>There are 4 DW needed to specify the Thread Group ID and the execution mask.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>6h</td><td>DWORD_COUNT_n <b>[Default]</b></td><td>Excludes DWord (0,1)</td></tr></table></td></tr>
<tr><td>1</td><td>31:8</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>7</td><td><b>Shared Local Memory Fixed Offset</b>
This bit, if set, specifies that the offset into the 64k Shared Local Memory for the current thread group is specified by software in the Shared Local Memory Offset field.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td>Thread Groups Offset</td><td>Offset to start of segment determined by hardware based on concurrently running thread groups.</td></tr><tr><td>1</td><td>Shared Local Memory Offset</td><td>Offset to start of the Shared Local Memory segment supplied in Shared Local Memory Offset</td></tr></table></td></tr>
<tr><td></td><td>6:5</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
</table>

<table>
<tr><th colspan="3">GPGPU_OBJECT</th></tr>
<tr><td></td><td>4:0</td><td><b>Interface Descriptor Offset</b>
<table><tr><td>Format:</td><td>U5</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. In VLD mode, this field is ignored by hardware.</td></tr>
<tr><td>2</td><td>31:28</td><td><b>Shared Local Memory Offset</b>
<table><tr><td>Format:</td><td>U4</td></tr></table>
If the Shared Local Memory Fixed Offset is set, this field provides the offset to the start of the Shared Local Memory for this thread group. The value of this field is multiplied by 4k to get the starting address. All threads in the thread group must have the same value.</td></tr>
<tr><td></td><td>27:25</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>24</td><td><b>End of Thread Group</b>
<table><tr><td>Format:</td><td>Boolean</td></tr></table>
This bit indicates that this dispatch is the last for the current thread group.</td></tr>
<tr><td></td><td>23:19</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>18:17</td><td><b>Half-Slice Destination Select</b>
This field selects the half slice that this thread must be sent to.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>00b</td><td>Either half-slice</td></tr>
<tr><td>01b</td><td>Half-Slice 0</td></tr>
<tr><td>10b</td><td>Half-Slice 1</td></tr>
</table></td></tr>
<tr><td></td><td>16:0</td><td><b>Indirect Data Length</b>
<table><tr><td>Format:</td><td>U17 in bytes</td></tr></table>
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.
DWord Length = 0</td></tr>
<tr><td>3</td><td>31:0</td><td><b>Indirect Data Start Address</b>
<table><tr><td>Format:</td><td>IndirectObjectBaseOffset[31:0]</td></tr></table>
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 <b>Indirect Object Base Address</b>.
Hardware ignores this field if indirect data is not present.
The start address is DWord aligned address.
(Bits 31:29 MBZ)
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>[0,512MB]</td><td></td></tr>
</table></td></tr>
</table>

GPGPU_OBJECT

4	31:0	**Thread Group ID X** This is the X coordinate of the group id.
5	31:0	**Thread Group ID Y** This is the Y coordinate of the group id for all channels generated by this command.
6	31:0	**Thread Group ID Z** This is the Z coordinate of the thread group id.
7	31:0	**Execution Mask** Format: Must Be All Ones Must be 0xFFFFFFFF This provides a bit per channel enable for the SIMD32 dispatch. The LSB of the Mask enables the execution of SIMD32 channel 0; the remaining bits enable the corresponding channel numbers. SIMD16 and SIMD8 dispatches should use the LSB bits of the mask. Any disabled channel will not read or write data to memory.

GPGPU_WALKER

Source: RenderCS
Length Bias: 2

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h GFXPIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Pipeline</b><table><tr><td>Default Value:</td><td>2h Media</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>26:24</td><td><b>Media Command Opcode</b><table><tr><td>Default Value:</td><td>1h GPGPU_WALKER</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>23:16</td><td><b>SubOpcode A</b><table><tr><td>Default Value:</td><td>05h GPGPU_WALKER SubOp</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:11</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>10</td><td><b>Indirect Parameter Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>If set, the values in DW 4, 6, 8 are ignored and replaced by the current values of the corresponding GPGPU_xxx MMIO registers:
GPGPU_DISPATCHDIMX (instead of DW4)
GPGPU_DISPATCHDIMY (instead of DW6)
GPGPU_DISPATCHDIMZ (instead of DW8)</td></tr>
<tr><td></td><td>9</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>8</td><td><b>Predicate Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>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><b>DWord Length</b><table><tr><td>Format:</td><td>=n</td></tr></table>Total Length - 2<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>9h</td><td>DWORD_COUNT_n <b>[Default]</b></td><td>Allowed value is 9</td></tr></table></td></tr>
<tr><td>1</td><td>31:8</td><td><b>Reserved</b></td></tr>
</table>

<table>
<tr><th colspan="3">GPGPU_WALKER</th></tr>
<tr><td></td><td>7:5</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>4:0</td><td><b>Interface Descriptor Offset</b><table><tr><td>Format:</td><td>U5</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>31:30</td><td><b>SIMD Size</b>
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></td></tr>
<tr><td></td><td>29:22</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>21:16</td><td><b>Thread Depth Counter Maximum</b>
The maximum value of the thread depth counter. Since the counter starts at 0, the depth is this value + 1. <b>(Thread_Depth_Max+1)*(Thread_Height_Max+1)*(Thread_Width_Max+1)</b> must equal <b>Number of Threads in GPGPU Thread Group</b> in the Interface Descriptor.</td></tr>
<tr><td></td><td>15:14</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>13:8</td><td><b>Thread Height Counter Maximum</b>
The maximum value of the thread height counter. The height is this value + 1.</td></tr>
<tr><td></td><td>7:6</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>5:0</td><td><b>Thread Width Counter Maximum</b>
The maximum value of the thread width counter. The height is this value + 1.</td></tr>
<tr><td>3</td><td>31:0</td><td><b>Thread Group ID Starting X</b>
This is the initial value of the X component of the thread group. When X reaches the maximum value it rolls around to 0, not to this value.</td></tr>
<tr><td>4</td><td>31:0</td><td><b>Thread Group ID X Dimension</b>
The X dimension of the thread group (maximum X is dimension -1)</td></tr>
<tr><td>5</td><td>31:0</td><td><b>Thread Group ID Starting Y</b>
This is the initial value of the Y component of the thread group. When Y reaches the maximum value it rolls around to 0, not to this value.</td></tr>
<tr><td>6</td><td>31:0</td><td><b>Thread Group ID Y Dimension</b>
The Y dimension of the thread group (maximum Y is dimension -1)</td></tr>
<tr><td>7</td><td>31:0</td><td><b>Thread Group ID Starting Z</b>
This is the initial value of the Z component of the thread group</td></tr>
</table>

GPGPU_WALKER

8	31:0	**Thread Group ID Z Dimension** The Z dimension of the thread group (maximum Z is dimension -1)
9	31:0	**Right Execution Mask** Format: Must Be All Ones Must be 0xFFFFFFFF **Programming Notes** When simulating SIMD64 to fit large thread groups this field must be 0xFFFFFFFF and the actual execution masks passed in the payload.
10	31:0	**Bottom Execution Mask** Format: Must Be All Ones Must be 0xFFFFFFFF **Programming Notes** When simulating SIMD64 to fit large thread groups this field must be 0xFFFFFFFF and the actual execution masks passed in the payload.

f16to32 - Half Precision Float to Single Precision Float

Source:	EuIsa
Length Bias:	4

The f16to32 instruction converts the half precision float in src0 to single precision float and storing in dst.

Because this instruction does not have a 16-bit floating-point type, the source data type must be Word (W). The destination type must be F (Float).

Format:
[(pred)] f16to32[.cmod] (exec_size) dst src0

Restriction
Restriction: The FP Mode (Single Precision Floating Point Mode in cr0) must be IEEE mode.
Restriction: No accumulator access, implicit or explicit.

Syntax

```
[(pred)] f16to32[.cmod] (exec_size) reg reg [(pred)] f16to32[.cmod] (exec_size) reg imm16
```

Pseudocode

```
Evaluate(WrEn); for ( n = 0; n < exec_size; n++ ) { if ( WrEn.chan[n] ) { dst.chan[n] =
convert half precision float to single precision float(src0.chan[n]); } }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types
W	F

DWord	Bit	Description
0..3	127:64	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32
	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

halt - Halt

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 16-bit instruction pointer offsets. Both the JIP and UIP are signed 16-bit numbers, added to IP pre-increment. In GEN binary, JIP and UIP are at location src1 and must be of type W (signed word integer).

Format:
[(pred)] halt (exec_size) JIP UIP

Restriction

Restriction: dst and src0 must be NULL.

Syntax

```
[(pred)] halt (exec_size) imm16 imm16
```

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	127:112	**UIP** Format: S15 The jump distance in number of eight-byte units if a jump is taken for the channel.
	111:96	**JIP** Format: S15 The jump distance in number of eight-byte units if a jump is taken for the instruction.

halt - Halt

	95:64	**Reserved**	
		Format:	MBZ
	63:32	**Operand Control**	
		Format:	EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header**	
		Format:	EU_INSTRUCTION_HEADER

if - If

Source:	EuIsa
Length Bias:	4

Description

An if instruction starts an if/endif or an if/else/endif block of code. It restricts execution within the conditional block to only those channels that were enabled via the predicate control.

Each if instruction must have a matching endif instruction and may have up to one matching else instruction before the matching endif.

If all channels are inactive (for the if/endif or if/else/endif block), a jump is performed to the instruction referenced by JIP. This jump must be to right after the matching else instruction when present, or otherwise to the matching endif instruction of the conditional 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 16-bit instruction pointer offsets. Both the JIP and UIP are signed 16-bit numbers, added to IP pre-increment. In GEN binary, JIP and UIP are at location src1 and must be of type W (signed word integer).

Format:
[(pred)] if (exec_size) JIP UIP

Restriction

Restriction: The execution size must be the same for the if, else, and endif instructions of the same code block.

Syntax

```
[(pred)] if (exec_size) imm16 imm16
```

Pseudocode

```
Evaluate(WrEn); for ( n = 0; n < 32; n++ ) { if ( WrEn.channel[n] == 0 ) { PcIP[n] = IP +
JIP; } else { PcIP[n] = IP + 1; } } if ( PcIP != (IP + 1) ) { // for all channels Jump(IP
+ JIP); }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	N	N

DWord	Bit	Description
0..3	127:112	**UIP** Format: S15 The jump distance in number of eight-byte units if a jump is taken for the channel.
	111:96	**JIP** Format: S15 The jump distance in number of eight-byte units if a jump is taken for the instruction.

if - If

	95:64	**Reserved** Format: MBZ
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

illegal - Illegal

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
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

subb - Integer Subtraction with Borrow

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

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	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM
	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

jmpi - Jump Indexed

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 -2 (if the jmpi instruction is in native format) or -1 (if the jmpi instruction is in compact format) is an infinite loop on the jmpi instruction.

Restriction
Restriction: The execution size must be 1.
Restriction: The {NoMask} instruction option must be specified.
Restriction: The index data type must be D (Signed DWord Integer).

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. }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

Src Types
D

jmpi - Jump Indexed

DWord	Bit	Description
0..3	127:112	**Reserved** Format: MBZ
	111:96	**JIP** Format: S15 Jump Target Offset. The relative offset in 64-bit units if a jump is taken for the instruction.
	95:91	**Reserved** Format: MBZ
	90	**Flag Register Number** Added a second flag register
	89	**Flag Subregister Number** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN1
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align16') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN16
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

lzd - Leading Zero Detection

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

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	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32
	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

line - Line

Source:	EuIsa
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
Restriction: This is a specialized instruction that only supports an execution size (ExecSize) of 8 or 16.
Restriction: The src0 region must be a replicated scalar (with HorzStride == VertStride == 0).
Restriction: src0 must specify .0 or .4 as the subregister number, corresponding to a subregister byte offset of 0 or 16.
Restriction: 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	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM
	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

lrp - Linear Interpolation

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
Restriction: The vertical stride (VertStride) is overloaded to 4 in HW for 3-source instructions.
Restriction: The overflow conditional modifier (.o) is not allowed.
Restriction: No explicit accumulator access because this is a three-source instruction. AccWrEn is allowed for implicitly updating the accumulator.
Restriction: All three-source instructions have certain restrictions, described in Instruction Machine Formats.

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
	105	**Reserved** Format: MBZ
	104:85	**Source 1** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC
	84	**Reserved** Format: MBZ
	83:64	**Source 0** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC

lrp - Linear Interpolation

<table>
<tr><td></td><td>63:56</td><td><b>Destination Register Number</b><table><tr><td>Format:</td><td>DstRegNum</td></tr></table></td></tr>
<tr><td></td><td>55:53</td><td><b>Destination Subregister Number</b><table><tr><td>Format:</td><td>DstSubRegNum[2:0]</td></tr></table></td></tr>
<tr><td></td><td>52:49</td><td><b>Destination Channel Enable</b><table><tr><td>Format:</td><td>ChanEn[4]</td></tr></table>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 <i>x</i>, <i>y</i>, <i>z</i>, and <i>w</i>, respectively, where <i>x</i> corresponds to Channel 0 in the group and <i>w</i> corresponds to channel 3 in the group</td></tr>
<tr><td></td><td>48</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>47</td><td><b>NibCtrl</b><table><tr><td>Format:</td><td>NibCtrl</td></tr></table></td></tr>
<tr><td></td><td>46</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>45:44</td><td><b>Destination Data Type</b>
This field contains the data type for the destination<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>Single Precision Float</td></tr><tr><td>01b</td><td>DWord</td></tr><tr><td>10b</td><td>Unsigned DWord</td></tr><tr><td>11b</td><td>Double Precision Float</td></tr></table></td></tr>
<tr><td></td><td>43:42</td><td><b>Source Data Type</b>
This field contains the data type for all three sources<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>Single Precision Float</td></tr><tr><td>01b</td><td>DWord</td></tr><tr><td>10b</td><td>Unsigned DWord</td></tr><tr><td>11b</td><td>Double Precision Float</td></tr></table></td></tr>
<tr><td></td><td>41:40</td><td><b>Source 2 Modifier</b><table><tr><td>Exists If:</td><td>([Property[Source Modification]=='true')</td></tr><tr><td>Format:</td><td>SrcMod</td></tr></table></td></tr>
<tr><td></td><td>39:38</td><td><b>Source 1 Modifier</b><table><tr><td>Exists If:</td><td>([Property[Source Modification]=='true')</td></tr><tr><td>Format:</td><td>SrcMod</td></tr></table></td></tr>
</table>

lrp - Linear Interpolation

	41:36	**Reserved** Exists If: ([Property[Source Modification]=='false') Format: MBZ
	37:36	**Source 0 Modifier** Exists If: ([Property[Source Modification]=='true') Format: SrcMod
	35	**Reserved** Format: MBZ
	34	**Flag Register Number** This field contains the flag register number for instructions with a non-zero Conditional Modifier.
	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

and - Logic And

Source:	EuIsa
Length Bias:	4

Description
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 numeric, optionally changing a source value s to -s, abs(s), or -abs(s) before the AND operation.
Format: Source modifier is not allowed if source is an accumulator.

Restriction
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
*B,*W,*D	*B,*W,*D

DWord	Bit	Description
0..3	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM
	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

not - Logic Not

Source:	EuIsa
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 numeric, optionally changing a source value s to -s, abs(s), or -abs(s) before the NOT operation.

Format:
[(pred)] not[.cmod] (exec_size) dst src0

Restriction

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	Y	Y

Src Types	Dst Types
*B,*W,*D	*B,*W,*D

DWord	Bit	Description
0..3	127:64	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32
	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

or - Logic Or

Source:	EuIsa
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 numeric, optionally changing a source value s to -s, abs(s), or -abs(s) before the OR operation.
Format: [(pred)] or[.cmod] (exec_size) dst src0 src1

Restriction
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	Y	Y

Src Types	Dst Types
*B,*W,*D	*B,*W,*D

DWord	Bit	Description
0..3	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM
	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

xor - Logic Xor

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 numeric, optionally changing a source value s to -s, abs(s), or -abs(s) before the XOR operation.
Format: [(pred)] xor[.cmod] (exec_size) dst src0 src1

Restriction
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	Y	Y

Src Types	Dst Types
*B,*W,*D	*B,*W,*D

DWord	Bit	Description
0..3	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM
	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

MEDIA_CURBE_LOAD

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: 0h MEDIA_CURBE_LOAD Format: OpCode
	23:16	**SubOpcode** Default Value: 1h MEDIA_CURBE_LOAD SubOp Format: OpCode
	15:0	**DWord Length** Format: =n Total Length - 2 Value: 2h Name: DWORD_COUNT_n **[Default]** Description: Excludes DWord (0,1)
1	31:0	**Reserved** Format: MBZ
2	31:17	**Reserved** Format: MBZ
	16:0	**CURBE Total Data Length** 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 DWord (32-byte) aligned.

<table>
<tr><th colspan="3">MEDIA_CURBE_LOAD</th></tr>
<tr><td>3</td><td>31:0</td><td><b>CURBE Data Start Address</b>
<table><tr><td>Format:</td><td>DynamicStateOffset[31:0] CURBE</td></tr></table>
<table><tr><th>Description</th></tr><tr><td>Specifies the 32-byte (DWord) aligned address of the CURBE data. This pointer is relative to the <b>Dynamics Base Address</b>.</td></tr></table>
<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,FFFFFFFFh]</td><td></td></tr></table>
<table><tr><th>Programming Notes</th></tr><tr><td>Driver must invalidate the vertex fetch cache thru the <b>VF(address based) Cache Invalidation Enable</b> thru a PIPE_CONTROL command prior to reusing the same graphics memory space.
VF cache invalidation must be done when any graphics memory space is reused within the same 64-byte cacheline.</td></tr></table>
</td></tr>
</table>

MEDIA_INTERFACE_DESCRIPTOR_LOAD

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.

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** Format: MBZ
2	31:17	**Reserved** Format: MBZ
	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 [32,1024] | [1,32] interface descriptor entries

MEDIA_INTERFACE_DESCRIPTOR_LOAD

3	31:0	**Interface Descriptor Data Start Address**

Format:	DynamicStateOffset[31:0]INTERFACE_DESCRIPTOR_DATA

Description
This bit specifies the 32-byte aligned address of the Interface Descriptor data. This pointer is relative to the Dynamics Base Address.

Value	Name
[0,FFFFFFFFh]	

Programming Notes
Driver must invalidate the vertex fetch cache thru the **VF(address based) Cache Invalidation Enable** thru a PIPE_CONTROL command prior to reusing the same graphics memory space. VF cache invalidation must be done when any graphics memory space is reused within the same 64-byte cacheline.

MEDIA_OBJECT

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 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	**Reserved** Format: MBZ
	4:0	**Interface Descriptor Offset** Format: U5 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: Enable Indicates that the root thread may send spawn messages to spawn child threads and/or

MEDIA_OBJECT

<table>
<tr><td></td><td></td><td>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.
<i>In order avoid deadlock, such dereference must be issued once and only once for each URB handle.</i></td></tr>
<tr><td></td><td>30:25</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>24</td><td><b>Thread Synchronization</b>
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></td><td>23</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>22</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>21</td><td><b>Use Scoreboard</b>
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</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>19</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>18:17</td><td><b>Half-Slice Destination Select</b>
This field selects the half slice that this thread must be sent to.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>10b</td><td>Half-Slice 1</td><td>Cannot be used in products without a Half-Slice 1.</td></tr><tr><td>01b</td><td>Half-Slice 0</td><td></td></tr><tr><td>00b</td><td>Either half-slice</td><td>Hardware will choose the slice based on load.</td></tr></table><table><tr><th>Programming Notes</th></tr><tr><td>If "Either half-slice" is selected then the Slice Destination Select must also specify "Either slice".</td></tr></table></td></tr>
</table>

MEDIA_OBJECT

<table>
<tr><td></td><td>16:0</td><td><b>Indirect Data Length</b>
<table><tr><td>Format:</td><td>U17 In bytes</td></tr></table>
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).</td></tr>
<tr><td>3</td><td>31:0</td><td><b>Indirect Data Start Address</b>
<table><tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr></table>
<table>
<tr><th>Description</th></tr>
<tr><td>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 <b>Indirect Object Base Address</b>.
Hardware ignores this field if indirect data is not present.
Alignment of this address depends on the mode of operation.</td></tr>
<tr><td>This field specifies the DWord aligned address of the indirect data.</td></tr>
</table>
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>[0,512MB]</td><td></td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>Driver must invalidate the vertex fetch cache through the VF(address based) Cache Invalidation Enable through a PIPE_CONTROL command prior to reusing the same graphics memory space.
VF cache invalidation must be done when any graphics memory space is reused within the same 64-byte cacheline.</td></tr>
<tr><td>Bits 31:29 MBZ</td></tr>
</table></td></tr>
<tr><td>4</td><td>31:25</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>24:16</td><td><b>Scoredboard Y</b>
<table><tr><td>Format:</td><td>U9</td></tr></table>
This field provides the Y term of the scoreboard value of the current thread.</td></tr>
<tr><td></td><td>15:9</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>8:0</td><td><b>Scoreboard X</b></td></tr>
</table>

MEDIA_OBJECT

		Format: U9 This field provides the X term of the scoreboard value of the current thread.
5	31:20	**Reserved** Format: MBZ
	19:16	**Scoreboard Color** Format: U4 This field specifies which dependency color the current thread belongs to. It affects the dependency scoreboard control.
	15:8	**Reserved** Format: MBZ
	7:0	**Scoreboard Mask** Format: Boolean Each bit indicates the corresponding dependency scoreboard is dependent on. This field is AND'd with the corresponding Scoreboard Mask field in the MEDIA_VFE_STATE command. **Bit n (for n = 0...7):** Scoreboard n is dependent, where bit 0 maps to n = 0.
6..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_PRT

Source:	RenderCS
Length Bias:	2

This command is for generating a Persistent Root Thread for the media pipeline. It only supports loading of inline data but not indirect data. 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** 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. Value: 0Eh — Name: DWORD_COUNT_n **[Default]** — Description: Excludes DWord (0,1)
1	31:6	**Reserved** Format: MBZ
	5	**Reserved** Format: MBZ
	4:0	**Interface Descriptor Offset** Format: U5 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: Enable 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.

<table>
<tr><th colspan="3">MEDIA_OBJECT_PRT</th></tr>
<tr><td></td><td></td><td>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><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>23</td><td><b>PRT_Fence Needed</b><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><b>PRT_FenceType</b>
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><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>3</td><td>31:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>4..15</td><td>31:0</td><td><b>Inline Data</b><table><tr><td>Format:</td><td>U32</td></tr></table></td></tr>
</table>

MEDIA_OBJECT_WALKER

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	**Reserved** Format: MBZ
	4:0	**Interface Descriptor Offset** Format: U5 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 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

MEDIA_OBJECT_WALKER

<table>
<tr><td></td><td></td><td>URB handle deference only when it receives a resource dereference message from the thread. <em>In order avoid deadlock, such dereference must be issued once and only once for each URB handle.</em></td></tr>
<tr><td></td><td>30:25</td><td><strong>Reserved</strong><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>24</td><td><strong>Thread Synchronization</strong>
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></td><td>23:22</td><td><strong>Reserved</strong><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>21</td><td><strong>Use Scoreboard</strong>
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>20:17</td><td><strong>Reserved</strong><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>16:0</td><td><strong>Indirect Data Length</strong><table><tr><td>Format:</td><td>U17 in bytes</td></tr></table>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).</td></tr>
<tr><td>3</td><td>31:0</td><td><strong>Indirect Data Start Address</strong><table><tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr></table><table><tr><th>Description</th></tr><tr><td>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 <strong>Indirect Object Base Address.</strong>
Hardware ignores this field if indirect data is not present.
Alignment of this address depends on the mode of operation.</td></tr><tr><td>It is the DWord aligned address of the indirect data.</td></tr></table></td></tr>
</table>

<table>
<tr><th colspan="3">MEDIA_OBJECT_WALKER</th></tr>
<tr><td></td><td></td><td><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>
<tr><td>4</td><td>31:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>5</td><td>7:0</td><td><b>Scoreboard Mask</b><table><tr><td>Format:</td><td>Boolean</td></tr></table>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.
<b>Bit n (for n = 0...7):</b> Scoreboard n is dependent, where bit 0 maps to n = 0.</td></tr>
<tr><td>6</td><td>31</td><td><b>Dual Mode</b><table><tr><td>Format:</td><td>Boolean</td></tr></table></td></tr>
<tr><td></td><td>30</td><td><b>Repel</b><table><tr><td>Format:</td><td>Boolean</td></tr></table><table><tr><th>Programming Notes</th></tr><tr><td>Repel should not be combined with either Dual Mode or Quad Mode.</td></tr></table></td></tr>
<tr><td></td><td>29</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>28</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>27:24</td><td><b>Color Count Minus One</b><table><tr><td>Format:</td><td>U4</td></tr></table>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.
<b>Usage Example:</b> This allows multiple sets of dependency threads to be dispatched.</td></tr>
<tr><td></td><td>23:21</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>20:16</td><td><b>Middle Loop Extra Steps</b><table><tr><td>Format:</td><td>U5</td></tr></table></td></tr>
<tr><td></td><td>15:14</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>13:12</td><td><b>Local Mid-Loop Unit Y</b><table><tr><td>Format:</td><td>S1</td></tr></table></td></tr>
<tr><td></td><td>11:10</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
</table>

MEDIA_OBJECT_WALKER

	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 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	**Local End Y**

MEDIA_OBJECT_WALKER

		Format: U9 Ending vertical position of the local loop.
	15:9	**Reserved** Format: MBZ
	8:0	**Local End X** Format: U9 Ending horizontal position of the local loop.
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
	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**

MEDIA_OBJECT_WALKER

		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 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

		MEDIA_OBJECT_WALKER
	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

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

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b>
<table>
<tr><td>Default Value:</td><td>3h GFXPIPE</td></tr>
<tr><td>Format:</td><td>OpCode</td></tr>
</table></td></tr>
<tr><td></td><td>28:27</td><td><b>Pipeline</b>
<table>
<tr><td>Default Value:</td><td>2h Media</td></tr>
<tr><td>Format:</td><td>OpCode</td></tr>
</table></td></tr>
<tr><td></td><td>26:24</td><td><b>Media Command Opcode</b>
<table>
<tr><td>Default Value:</td><td>0h MEDIA_STATE_FLUSH</td></tr>
<tr><td>Format:</td><td>OpCode</td></tr>
</table></td></tr>
<tr><td></td><td>23:16</td><td><b>SubOpcode A</b>
<table>
<tr><td>Default Value:</td><td>4h MEDIA_STATE_FLUSH SubOp</td></tr>
<tr><td>Format:</td><td>OpCode</td></tr>
</table></td></tr>
<tr><td></td><td>15:0</td><td><b>DWord Length</b>
<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>0h</td><td>DWORD_COUNT_n <b>[Default]</b></td><td>Excludes DWord (0,1)</td></tr>
</table></td></tr>
<tr><td>1</td><td>31:9</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>8</td><td><b>Disable Pre-emption</b>
<table>
<tr><td>Format:</td><td>Enable</td></tr>
</table>
This bit causes the video front-end to ignore pre-emption requests if set. If this bit is set then</td></tr>
</table>

MEDIA_STATE_FLUSH

<table>
<tr><td></td><td></td><td>ARB_CHECK commands should not be used with it.
A subsequent MEDIA_STATE_FLUSH command with this bit cleared will honor previous pre-emption requests.</td></tr>
<tr><td></td><td>7</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>6</td><td><b>Watermark Required</b>
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.<table><tr><th>Programming Notes</th></tr><tr><td>If pre-emption is used, the WatermarkRequired bit must not be set.</td></tr></table></td></tr>
<tr><td></td><td>5:0</td><td><b>Interface Descriptor Offset</b><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 describes what resources are required to meet the watermark.</td></tr>
</table>

MEDIA_VFE_STATE

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:	0h MEDIA_VFE_STATE	
		Format:	OpCode	
	23:16	**SubOpcode A**		
		Default Value:	0h MEDIA_VFE_STATE SubOp	
		Format:	OpCode	
	15:0	**DWord Length**		
		Format:	=n Total Length - 2	
		Value	**Name**	**Description**
		06h	DWORD_COUNT_n **[Default]**	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.**		
	9:4	**Reserved**		
		Format:	MBZ	

MEDIA_VFE_STATE

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	Name	Description
[0,11]		indicating [1k bytes, 12k bytes]

Programming Notes
The definition of this field is different from that in 3D fixed functions, where the per-thread scratch space is specified in powers of 2.

2 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.

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	Name	Description
[0,64]		[0,64] Entries

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**

This field configures Gateway to use a simple message protocol.

Value	Name
0h	Maintaining OpenGateway/ForwardMsg/CloseGateway protocol (legacy mode)

<table>
<tr><th colspan="3">MEDIA_VFE_STATE</th></tr>
<tr><td></td><td></td><td><table><tr><td>1h</td><td>Bypassing OpenGateway/CloseGateway protocol</td></tr></table></td></tr>
<tr><td></td><td>5</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>4:3</td><td><b>Gateway MMIO Access Control</b>
The Gateway allows messages from EUs to read and write MMIO registers. This field limits this feature for security reasons<table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>No MMIO read/write allowed</td></tr><tr><td>1</td><td>Reserved</td></tr><tr><td>2</td><td>MMIO read/write to any address</td></tr></table></td></tr>
<tr><td></td><td>2</td><td><b>GPGPU Mode</b>
This bit indicates whether the VFE is in GPGPU mode (will expect GPGPU_OBJECT and GPGPU_WALKER commands) or MEDIA mode (will expect MEDIA_OBJECT and MEDIA_WALKER commands)<table><tr><th>Value</th><th>Name</th></tr><tr><td>0h</td><td>MEDIA Mode</td></tr><tr><td>1h</td><td>GPGPU Mode</td></tr></table></td></tr>
<tr><td></td><td>1:0</td><td><b>Reserved</b></td></tr>
<tr><td>3</td><td>31:8</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>7:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>4</td><td>31:16</td><td><b>URB Entry Allocation Size</b><table><tr><td>Format:</td><td>U16-1</td></tr></table><table><tr><td>Specifies the length of each URB entry used by the unit, in 256-bit register increments - 1. ROB address for URB starts after CURBE Allocated region.
(URB Entry Allocation Size * Number of URB Entries) + CURBE Allocation Size + 32) must be less than or equal to the number of entries in the URB as described in vol5c.5 Shared Functions Unified Return Buffer, under the section "URB Size".</td></tr><tr><td>If SLM is enabled then the number of available entries will be 1/3 the maximum URB entries.</td></tr></table><table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>[0,3040]</td><td></td><td>URB Entry Alloc Size when SLM is disabled.</td></tr><tr><td>[0,992]</td><td></td><td>URB Entry Alloc Size when SLM is enabled.</td></tr></table><table><tr><th>Programming Notes</th></tr><tr><td>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 then the allocation size does not matter, but the total Allocation Size * Number of URB Entries should be sufficient for the Indirect data.
If both Inline and Indirect are being used, then the allocation size must match the Inline and the total space must be enough for both the Indirect and Inline.</td></tr></table></td></tr>
</table>

<table>
<tr><th colspan="3">MEDIA_VFE_STATE</th></tr>
<tr><td></td><td>15:0</td><td><b>CURBE Allocation Size</b>
<table><tr><td>Format:</td><td>U12-1</td></tr></table>
<table><tr><td>Specifies the total length allocated for CURBE, in 256-bit register increments - 1. ROB address for CURBE starts at address 32.
(URB Entry Allocation Size * Number of URB Entries) + CURBE Allocation Size + 32) must be less than or equal to the number of entries in the URB as described in vol5c.5 Shared Functions Unified Return Buffer, under the section "URB Size".</td></tr><tr><td>If SLM is enabled then the number of available entries will be 1/3 the maximum URB entries.</td></tr></table>
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>[0,2016]</td><td></td><td>CURBE Alloc Size when SLM is disabled</td></tr><tr><td>[0,992]</td><td></td><td>CURBE Alloc Size when SLM is enabled</td></tr></table></td></tr>
<tr><td>5</td><td>31</td><td><b>Scoreboard Enable</b>
<table><tr><td>This field enables and disables the hardware scoreboard in the Media Pipeline. If this field is cleared, hardware ignores the following scoreboard state fields.</td></tr></table>
<table><tr><th>Value</th><th>Name</th></tr><tr><td>0h</td><td>Scoreboard disabled</td></tr><tr><td>1h</td><td>Scoreboard enabled</td></tr></table></td></tr>
<tr><td></td><td>30</td><td><b>Scoreboard Type</b>
This field selects the type of scoreboard in use.
<table><tr><th>Value</th><th>Name</th></tr><tr><td>0h</td><td>Stalling scoreboard</td></tr><tr><td>1h</td><td>Non-stalling scoreboard</td></tr></table></td></tr>
<tr><td></td><td>29:8</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>Scoreboard Mask</b>
<table><tr><td>Format:</td><td>Enable[8]</td></tr></table>
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.</td></tr>
<tr><td>6</td><td>31:28</td><td><b>Scoreboard 3 Delta Y</b>
<table><tr><td>Format:</td><td>S3</td></tr></table>
Relative vertical distance of the dependent instance assigned to scoreboard 3, in the form of 2's compliment.</td></tr>
<tr><td></td><td>27:24</td><td><b>Scoreboard 3 Delta X</b>
<table><tr><td>Format:</td><td>S3</td></tr></table>
Relative horizontal distance of the dependent instance assigned to scoreboard 3, in the form of 2's compliment.</td></tr>
</table>

MEDIA_VFE_STATE

	23:20	**Scoreboard 2 Delta Y** Format: | S3 Relative vertical distance of the dependent instance assigned to scoreboard 2, in the form of 2's compliment.
	19:16	**Scoreboard 2 Delta X** Format: | S3 Relative horizontal distance of the dependent instance assigned to scoreboard 2, in the form of 2's compliment.
	15:12	**Scoreboard 1 Delta Y** Format: | S3 Relative vertical distance of the dependent instance assigned to scoreboard 1, in the form of 2's compliment.
	11:8	**Scoreboard 1 Delta X** Format: | S3 Relative horizontal distance of the dependent instance assigned to scoreboard 1, in the form of 2's compliment.
	7:4	**Scoreboard 0 Delta Y** Format: | S3 Relative vertical distance of the dependent instance assigned to scoreboard 0, in the form of 2's compliment.
	3:0	**Scoreboard 0 Delta X** Format: | S3 Relative horizontal distance of the dependent instance assigned to scoreboard 0, in the form of 2's compliment.
7	31:28	**Scoreboard 7 Delta Y** Format: | S3 Relative vertical distance of the dependent instance assigned to scoreboard 7, in the form of 2's compliment.
	27:24	**Scoreboard 7 Delta X** Format: | S3 Relative horizontal distance of the dependent instance assigned to scoreboard 7, in the form of 2's compliment.
	23:20	**Scoreboard 6 Delta Y**

MEDIA_VFE_STATE

		Format: S3 Relative vertical distance of the dependent instance assigned to scoreboard 6, in the form of 2's compliment.
	19:16	**Scoreboard 6 Delta X** Format: S3 Relative horizontal distance of the dependent instance assigned to scoreboard 6, in the form of 2's compliment.
	15:12	**Scoreboard 5 Delta Y** Format: S3 Relative vertical distance of the dependent instance assigned to scoreboard 5, in the form of 2's compliment.
	11:8	**Scoreboard 5 Delta X** Format: S3 Relative horizontal distance of the dependent instance assigned to scoreboard 5, in the form of 2's compliment.
	7:4	**Scoreboard 4 Delta Y** Format: S3 Relative vertical distance of the dependent instance assigned to scoreboard 4, in the form of 2's compliment.
	3:0	**Scoreboard 4 Delta X** Format: S3 Relative horizontal distance of the dependent instance assigned to scoreboard 4, in the form of 2's compliment.

MFC_AVC_PAK_OBJECT

Source:	VideoCS
Length Bias:	2

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFC_AVC_PAK_OBJECT Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h AVC_ENC Format: OpCode
	23:21	**SubOpcode A** Default Value: 2h Format: OpCode
	20:16	**SubOpcode B** Default Value: 9h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Length -2 Value | Name 0009h | DWORD_COUNT_n **[Default]**
1	31:10	**Reserved** Format: MBZ
	9:0	**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

MFC_AVC_PAK_OBJECT

<table>
<tr><td></td><td></td><td>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>2</td><td>31:29</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>28:0</td><td><b>Indirect PAK-MV Data Start Address Offset</b>
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><b>Inline Data</b>
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_MPEG2_PAK_OBJECT

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** Format: MBZ
	11:0	**DWord Length** Default Value: 0007h Excludes DWord (0,1) Format: =n Total Length - 2
1..8	31:0	**Inline Data** All the required MB level controls and parameters for encoding are captured as inline data of

MFC_MPEG2_PAK_OBJECT		
		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

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).

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h PARALLEL_VIDEO_PIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Pipeline</b><table><tr><td>Default Value:</td><td>2h MFX_MPEG2_SLICEGROUP_STATE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>26:24</td><td><b>Media Command Opcode</b><table><tr><td>Default Value:</td><td>3h MPEG2</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>23:21</td><td><b>SubOpcode A</b><table><tr><td>Default Value:</td><td>2h MEDIA_</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>20:16</td><td><b>SubOpcode B</b><table><tr><td>Default Value:</td><td>3h MEDIA_</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:12</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>6h Excludes DWord (0,1)</td></tr><tr><td>Format:</td><td>=n Total Length - 2</td></tr></table></td></tr>
<tr><td>1</td><td>31</td><td><b>MbRateCtrlFlag- RateControlCounterEnable (Encoder-only)</b>
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<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>
<tr><td></td><td>30</td><td><b>MbRateCtrlReset- ResetRateControlCounter (Encoder-only)</b>
To reset the bit allocation accumulation counter to 0 to restart the rate control.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Disable</td><td>Not reset</td></tr><tr><td>1h</td><td>Enable</td><td>reset</td></tr></table></td></tr>
</table>

MFC_MPEG2_SLICEGROUP_STATE

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
0	Disable
1	Enable

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**

Format:	MBZ

20 **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

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

19 **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.

18 **Reserved**

<table>
<tr><th colspan="3">MFC_MPEG2_SLICEGROUP_STATE</th></tr>
<tr><td></td><td>17</td><td><b>HeaderPresentFlag - Header Insertion Present in Bitstream (Encoder-only)</b>
<table>
<tr><th>Value</th><th>Name</th><th>Description</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></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></tr>
</table></td></tr>
<tr><td></td><td>16</td><td><b>SliceData PresentFlag - SliceData Insertion Present in Bitstream (Encoder-only)</b>
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Disable</td><td>no Slice Data insertion into the output bitstream buffer</td></tr>
<tr><td>1h</td><td>Enable</td><td>Slice Data insertion into the output bitstream buffer is present.</td></tr>
</table></td></tr>
<tr><td></td><td>15</td><td><b>TailPresentFlag - Tail Insertion Present in bitstream (Encoder-only)</b>
<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><b>FirstSliceHdrDisabled</b>
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><b>IntraSlice</b>
intra slice value included in slice headers, when IntraSliceFlag = 1.</td></tr>
<tr><td></td><td>12</td><td><b>IntraSliceFlag</b>
intra slice flag included in slice headers</td></tr>
<tr><td></td><td>11:8</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ for SliceID extension</td></tr>
</table></td></tr>
<tr><td></td><td>7:4</td><td><b>SliceID[3:0] (Encoder-only)</b>
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><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ for StreamID extension</td></tr>
</table></td></tr>
<tr><td></td><td>1:0</td><td><b>StreamID[1:0] (Encoder-only)</b>
To identify the output data (coding information record) returned for rate control from PAK to ENC and VPP</td></tr>
<tr><td>2</td><td>31:24</td><td><b>NextSgMbYcnt - also NextStartVertPos</b>
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.</td></tr>
<tr><td></td><td>23:16</td><td><b>NextSgMbXcnt - also NextStartHorzPos</b>
BitFieldDesc</td></tr>
<tr><td></td><td>15:8</td><td><b>FirstMbYcnt - also CurrStartVertPos</b>
<table>
<tr><td>Format:</td><td>U8</td></tr>
</table>
also CurrStartVertPos, Vertical count of the first MB in the current slice group (Encoder-only)</td></tr>
</table>

MFC_MPEG2_SLICEGROUP_STATE

<table>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td>7:0</td><td><b>FirstMbXcnt - also CurrStartHorzPos</b><table><tr><td>Format:</td><td>U8</td></tr></table>Horizontal count of the first MB in the current slice group (Encoder-only)</td></tr>
<tr><td>3</td><td>31:9</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>8</td><td><b>SliceGroupSkip</b><table><tr><td>Exists If:</td><td>//Encoder Only</td></tr><tr><td>Format:</td><td>U1</td></tr></table>All macroblocks are skipped</td></tr>
<tr><td></td><td>7:6</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>5:0</td><td><b>SliceGroupQp</b><table><tr><td>Exists If:</td><td>//Encoder Only</td></tr><tr><td>Format:</td><td>U6</td></tr></table>Initial slice quality parameter</td></tr>
<tr><td>4</td><td>31:29</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>28:0</td><td><b>BitstreamOffset - Indirect PAK-BSE Data Start Address (Write)</b><table><tr><td>Exists If:</td><td>//Encoder Only</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. This field is only valid for AVC encode mode.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,512MB)</td><td></td></tr></table></td></tr>
<tr><td>5</td><td>31:24</td><td><b>MaxQpNegModifier - Magnitude of QP Max Negative Modifier (Encoder-only)</b><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></td><td>23:16</td><td><b>MaxQpPosModifier - Magnitude of QP Max Positive Modifier (Encoder-only)</b><table><tr><td>Format:</td><td>U8</td></tr></table>This field specifies the upper limit of the QP modifier.</td></tr>
</table>

MFC_MPEG2_SLICEGROUP_STATE

<table>
<tr><td></td><td></td><td><table><tr><th>Value</th><th>Name</th></tr><tr><td>[0, 51]</td><td></td></tr></table></td></tr>
<tr><td></td><td>15:12</td><td><b>ShrinkParam - Shrink Resistance (Encoder-only)</b><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></td><td>11:8</td><td><b>Shrinkaram - Shrink Init (Encoder-only)</b><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></td><td>7:4</td><td><b>GrowParam - Grow Resistance (Encoder-only)</b><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></td><td>3:0</td><td><b>GrowParam - Grow Init (Encoder-only)</b><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>6</td><td>31:24</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>23:20</td><td><b>CorrectPoints - Correct 6 (Encoder-only)</b><table><tr><td>Format:</td><td>U4</td></tr></table>This field specifies the points used in the lowermost RC region when sum_act <= sum_min.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0, 15]</td><td></td></tr></table></td></tr>
<tr><td></td><td>19:16</td><td><b>CorrectPoints - Correct 5 (Encoder-only)</b><table><tr><td>Format:</td><td>U4</td></tr></table>This field specifies the points used in the fifth RC region when sum_act > sum_min but <= lower_midpt.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0, 15]</td><td></td></tr></table></td></tr>
<tr><td></td><td>15:12</td><td><b>CorrectPoints - Correct 4 (Encoder-only)</b><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.</td></tr>
</table>

<table>
<tr><th colspan="3">MFC_MPEG2_SLICEGROUP_STATE</th></tr>
<tr><td></td><td></td><td><table><tr><th>Value</th><th>Name</th></tr><tr><td>[0, 15]</td><td></td></tr></table></td></tr>
<tr><td></td><td>11:8</td><td><b>CorrectPoints - Correct 3 (Encoder-only)</b><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></td></tr>
<tr><td></td><td>7:4</td><td><b>CorrectPoints - Correct 2 (Encoder-only)</b><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></td></tr>
<tr><td></td><td>3:0</td><td><b>CorrectPoints - Correct 1 (Encoder-only)</b><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></td></tr>
<tr><td>7</td><td>31:28</td><td><b>CV7 - Clamp Value 7 (Encoder-only)</b><table><tr><td>Exists If:</td><td>//Encoder Only</td></tr></table></td></tr>
<tr><td></td><td>27:24</td><td><b>CV6 - Clamp Value 6 (Encoder-only)</b><table><tr><td>Exists If:</td><td>//Encoder Only</td></tr><tr><td>Format:</td><td>U4</td></tr></table></td></tr>
<tr><td></td><td>23:20</td><td><b>CV5 - Clamp Value 5 (Encoder-only)</b><table><tr><td>Exists If:</td><td>//Encoder Only</td></tr><tr><td>Format:</td><td>U4</td></tr></table></td></tr>
<tr><td></td><td>19:16</td><td><b>CV4 - Clamp Value 4 (Encoder-only)</b><table><tr><td>Exists If:</td><td>//Encoder Only</td></tr><tr><td>Format:</td><td>U4</td></tr></table></td></tr>
<tr><td></td><td>15:12</td><td><b>CV3 - Clamp Value 3 (Encoder-only)</b><table><tr><td>Exists If:</td><td>//Encoder Only</td></tr><tr><td>Format:</td><td>U4</td></tr></table></td></tr>
<tr><td></td><td>11:8</td><td><b>CV2 - Clamp Value 2 (Encoder-only)</b><table><tr><td>Exists If:</td><td>//Encoder Only</td></tr><tr><td>Format:</td><td>U4</td></tr></table></td></tr>
<tr><td></td><td>7:4</td><td><b>CV1 - Clamp Value 1 (Encoder-only)</b><table><tr><td>Exists If:</td><td>//Encoder Only</td></tr></table></td></tr>
</table>

MFC_MPEG2_SLICEGROUP_STATE

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
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

Source: VideoCS
Length Bias: 2

The MFD_AVC_BSD_OBJECT command is the only primitive command for the AVC Decoding Pipeline. The same command is used for both CABAC and CAVLD modes.
 The Slice Data portion of the bitstream is loaded as indirect data object.Before issuing a MFD_AVC_BSD_OBJECT command, all AVC states of the MFD Engine need to be valid. Therefore the commands used to set these states need to have been issued prior to the issue of a MFD_AVC_BSD_OBJECT command.

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** Default Value: 4h Excludes DWord (0,1) = 0004 Format: =n Total Length - 2
1	31:24	**Reserved** Format: MBZ
	23:0	**Indirect BSD Data Length** Format: U24 This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. This field must have the same alignment as the Indirect Object Data Start Address. AVC Short Format: It is the length in bytes of the bitstream data for the current slice, including Slice Header + Slice Data + Emulation Prevention Bytes + any filling trailing zeros after the last MB.

<table>
<tr><th colspan="3">MFD_AVC_BSD_OBJECT</th></tr>
<tr><td></td><td></td><td>Hardware ignores the contents after the last non-zero byte. Trailing zero is allowed and handled correctly in both CABAC and CAVLC modes.</td></tr>
<tr><td>2</td><td>31:29</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>28:0</td><td><b>Indirect BSD Data Start Address</b>
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 <b>MFD Indirect Object Base Address</b>.
Hardware ignores this field if indirect data is not present.
It is a byte-aligned address for the AVC bitstream data in both CABAC/CAVLD Modes.
In implementing a phantom slice at the end of a picture for automatic error concealment, this field should set to 0.
It includes the NAL Header (the NAL Header does not need to perform EMU detection). For AVC Base Layer, it is a single byte. But for MVC, the NAL Header is 4 Bytes long. These NAL Header Unit must be passed to HW in the compressed bitstream buffer.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,512MB)</td><td></td></tr></table></td></tr>
<tr><td>3..5</td><td>31:0</td><td><b>Inline Data</b>
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.</td></tr>
</table>

MFD_AVC_DPB_STATE

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 this implementation's 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 this implementation's POCList[34] of MFX_AVC_DIRECTMODE_STATE command.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h PARALLEL_VIDEO_PIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Pipeline</b><table><tr><td>Default Value:</td><td>2h MFX_MULTI_DW</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>26:24</td><td><b>Media Command Opcode</b><table><tr><td>Default Value:</td><td>1h AVC_DEC</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>23:21</td><td><b>SubOpcode A</b><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>20:16</td><td><b>SubOpcode B</b><table><tr><td>Default Value:</td><td>6h</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:12</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11:0</td><td><b>DWord Length</b><table><tr><td>Format:</td><td>=n Total Length - 2</td></tr></table><table><tr><th>Value</th><th>Name</th></tr><tr><td>9h</td><td>Excludes DWord (0,1) <b>[Default]</b></td></tr></table></td></tr>
<tr><td>1</td><td>31:16</td><td><b>LongTermFrame_Flag[16][1 bit]</b>
One-to-one correspondence with the entries of the this implementation's RefFrameList[16]. 1 bit per reference frame.<table><tr><th>Value</th><th>Name</th></tr><tr><td>1</td><td>the picture is a long term reference picture</td></tr><tr><td>0</td><td>the picture is a short term reference picture</td></tr></table></td></tr>
<tr><td></td><td>15:0</td><td><b>Non-ExistingFrame_Flag[16][1 bit]</b>
One-to-one correspondence with the entries of the this implementation's RefFrameList[16]. 1 bit</td></tr>
</table>

MFD_AVC_DPB_STATE

<table>
<tr><td></td><td></td><td>per reference frame.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>1</td><td>INVALID</td><td>the reference picture in that entry of RefFrameList[] does not exist anymore.</td></tr>
<tr><td>0</td><td>VALID</td><td>the reference picture in that entry of RefFrameList[] is a valid reference</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>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.</td></tr>
</table>
</td></tr>
<tr><td>2</td><td>31:0</td><td><b>UsedForReference_Flag[16][2 bits]</b>
One-to-one correspondence with the entries of the this implementation's RefFrameList[16]. 2 bits per reference frame.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>NOT_REFERENCE</td><td>indicates a frame is "not used for reference".</td></tr>
<tr><td>1</td><td>TOP_FIELD</td><td>bit[0] indicates that the top field of a frame is marked as "used for reference".</td></tr>
<tr><td>2</td><td>BOTTOM_FIELD</td><td>bit[1] indicates that the bottom field of a frame is marked as "used for reference".</td></tr>
<tr><td>3</td><td>FRAME</td><td>bit[1:0] indicates that a frame (or field pair) is marked as "used for reference".</td></tr>
</table>
</td></tr>
<tr><td>3..10</td><td>31:0</td><td><b>LTSTFrameNumList[16][16 bits]</b>
One-to-one correspondence with the entries of the this implementation's RefFrameList[16]. 16 bits per reference frame. Depending on the corresponding LongTermFrame_Flag[], the content of this field is interpreted differently.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>1</td><td>LongTermFrame_Flag[i]</td><td>LTSTFrameNumList[i] represent LongTermFrameIdx.</td></tr>
<tr><td>0</td><td>LongTermFrame_Flag[i]</td><td>LTSTFrameNumList[i]represent Short Term Picture FrameNum.</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>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.</td></tr>
</table>
</td></tr>
</table>

MFD_AVC_SLICEADDR

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
	11:0	**DWord Length** Default Value: 1h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:24	**Reserved** Format: MBZ

MFD_AVC_SLICEADDR

	23:0	**Indirect BSD Data Length** Format: | U24 in bytes This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. Driver always programs this up to 5 bytes; for bitstream less than 5 bytes, driver program the lesser value. (Emulation Prevention Byte should never happen for the first 5 bytes when the max picture size can only be 4Kx4K)It is the length in bytes of the bitstream data for the current slice, including Slice Header + Slice Data + Emulation Prevention Bytes + any filling trailing zeros after the last MB. Hardware ignores the contents after the last non-zero byte. Trailing zero is allowed and handled correctly in both CABAC and CAVLC modes.
2	31:29	**Reserved** 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** | **Name** [0,512MB) |

MFD_IT_OBJECT

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. 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**

MFD_IT_OBJECT

<table>
<tr><td></td><td></td><td><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>28:0</td><td><b>Indirect IT-MV Data Start Address Offset</b>
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).<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,512MB)</td><td></td></tr></table></td></tr>
<tr><td>3</td><td>31:12</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11:0</td><td><b>Indirect IT-COEFF Data Length</b>
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.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,3072]</td><td>In bytes [0, 256*3*4]</td></tr></table></td></tr>
<tr><td>4</td><td>31:29</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>28:0</td><td><b>Indirect IT-COEFF Data Start Address Offset</b>
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.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,512MB)</td><td></td></tr></table></td></tr>
<tr><td>5</td><td>31:6</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>5:0</td><td><b>Indirect IT-DBLK Control Data Length</b><table><tr><td>Format:</td><td>U6</td></tr></table>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</td></tr>
</table>

<table>
<tr><th colspan="3">MFD_IT_OBJECT</th></tr>
<tr><td></td><td></td><td>field is only valid in AVC decoder IT mode.</td></tr>
<tr><td>6</td><td>31:29</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>28:0</td><td><b>Indirect IT-DBLK Control Data Start Address Offset</b><table><tr><td>Format:</td><td>IndirectObjectBaseAddress[28:0]</td></tr></table>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.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,512MB)</td><td></td></tr></table></td></tr>
<tr><td>7..n</td><td>31:0</td><td><b>Inline Data</b>
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.</td></tr>
</table>

MFD_JPEG_BSD_OBJECT

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** Format: MBZ
	11:0	**DWord Length** Default Value: 004h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:0	**Indirect Data Length** . It is the length in bytes of the bitstream data for the current Scan. It includes the first byte of the first MCU and the last non-zero byte of the last MCU in the Scan. Specifically, the zero-padding bytes (if present) are excluded. Hardware ignores the contents after the last non-zero byte.
2	31:29	**Reserved** Format: MBZ
	28:0	**Indirect Data Start Address** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This pointer is relative to the BSD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the JPEG bitstream data
3	31:29	**Reserved** Format: MBZ
	28:16	**Scan Horizontal Position** Format: U13 bits in blocks This field indicates the horizontal position (in block units) of the first MCU in the Scan.

MFD_JPEG_BSD_OBJECT

<table>
<tr><th></th><th></th><th></th></tr>
<tr><td></td><td>15:13</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>12:0</td><td><b>Scan Vertical Position</b><table><tr><td>Format:</td><td>U13 bits in blocks</td></tr></table>This field indicates the vertical position (in block units) of the first MCU in the Scan.</td></tr>
<tr><td>4</td><td>31</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>30</td><td><b>Interleaved</b><table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td>Non-Interleaved</td><td>one component in the Scan</td></tr><tr><td>1</td><td>Interleaved</td><td>multiple components in the Scan</td></tr></table></td></tr>
<tr><td></td><td>29:27</td><td><b>Scan Components</b>
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.</td></tr>
<tr><td></td><td>26</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>25:0</td><td><b>MCU Count</b><table><tr><td>Format:</td><td>U26</td></tr></table>This field indicates the number of MCUs in the Scan.</td></tr>
<tr><td>5</td><td>31:16</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>15:0</td><td><b>RestartInterval(16 bit)</b><table><tr><td>Format:</td><td>U16</td></tr></table>Specifies the number of MCU in restart interval. Valid values are 1->0xFFFFValue of 0 implies that all the SCAN have only one ECS.</td></tr>
</table>

MFD_MPEG2_BSD_OBJECT

Source:	VideoCS
Length Bias:	2

Different from AVC and VC1, MFD_MPEG2_BSD_OBJECT command is pipelinable. This is for performance purpose as in MPEG2 a slice is defined as a group of MBs of any size that must be within a macroblock row. Slice header parameters are passed in as inline data and the bitstream data for the slice is passed in as indirect data. Of the inline data, slice_horizontal_position and slice_vertical_position determines the location within the destination picture of the first macroblock in the slice. The content in this command is identical to that in the MEDIA_OBJECT command in VLD mode described in the Media Chapter.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: | 3h PARALLEL_VIDEO_PIPE Format: | OpCode
	28:27	**Pipeline** Default Value: | 2h MFD_MPEG2_BSD_OBJECT Format: | OpCode
	26:24	**Media Command Opcode** Default Value: | 3h MPEG2_DEC Format: | OpCode
	23:21	**SubOpcode A** Default Value: | 1h Format: | OpCode
	20:16	**SubOpcode B** Default Value: | 8h Format: | OpCode
	15:12	**Reserved** Format: | MBZ
	11:0	**DWord Length** Default Value: | 0003h Excludes DWord (0,1) Format: | =n Total Length - 2
1	31:0	**Indirect BSD Data Length** 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** As MPEG-2 spec does not post any limitation of the size of zero-padding bytes, it is

MFD_MPEG2_BSD_OBJECT

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

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** Format: MBZ
	11:0	**DWord Length** Default Value: 0003h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:24	**Reserved** Format: MBZ
	23:0	**Indirect BSD Data Length** Format: U24 This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. This field must have the same alignment as the Indirect Object Data Start Address. Long Format: It is the length in bytes of the bitstream data for the current slice/picture. It includes the first byte of the

<table>
<tr><th colspan="3">MFD_VC1_BSD_OBJECT</th></tr>
<tr><td></td><td></td><td>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 Offset 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.</td></tr>
<tr><td>2</td><td>31:29</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>28:0</td><td><b>Indirect Data Start Address</b><table><tr><td>Format:</td><td>GraphicsAddress[28:0]</td></tr></table><table><tr><td>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.</td></tr></table><table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,512MB)</td><td></td></tr></table></td></tr>
<tr><td>3</td><td>31:24</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>23:16</td><td><b>Slice Start Vertical Position</b>
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.</td></tr>
<tr><td></td><td>15:9</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>8:0</td><td><b>Next Slice Vertical Position</b>
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.</td></tr>
<tr><td>4</td><td>31:16</td><td><b>First_MB_Byte_Offset of Slice Data or Slice Header</b>
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><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>4</td><td><b>Emulation Prevention Byte Present</b><table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td></td><td>H/W needs to perform Emulation Byte Removal</td></tr></table></td></tr>
</table>

MFD_VC1_BSD_OBJECT

<table>
<tr><td></td><td></td><td><table><tr><td>1h</td><td></td><td>H/W does not need to perform Emulation Byte Removal</td></tr></table></td></tr>
<tr><td></td><td>3</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>2:0</td><td><b>FirstMbBitOffset (First Macroblock Bit Offset)</b><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

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. Driver will need to perform addition operations to generate all the fields in this command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFD_VC1_LONG_PIC_STATE Format: OpCode
	26:24	**Media Command Opcode** Default Value: 2h VC1_DEC Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h Format: OpCode
	20:16	**SubOpcode B** Default Value: 1h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 0004h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:24	**Reserved** Format: MBZ
	23:16	**PictureHeightInMBsMinus1 (Picture Height Minus 1 in Macroblocks)** Format: U8 This field indicates the height of the picture in unit of macroblocks. For example, for a 1920x1080 frame picture, PictureHeightInMBs equals 68 (1080 divided by 16, and rounded up, i.e. effectively specified as 1088 instead). This field is used in VLD and IT modes. **Value** | **Name** | **Description**

MFD_VC1_LONG_PIC_STATE

[0,255]		a valid range of [0,255] [1, 256] MB

Programming Notes
Note: Even though the Advanced Profile allows frame dimensions (width, height) to not be aligned to macroblock boudary, it doesn't affect the bitstream decoding. And it is preferable to use 'intermediate buffer' that is macroblock aligned for decoding. In order to simplify the out-of-bound reference pixel access, the out-of-bound extrapolation rule in VC1 spec can be used to expand the expected decoded frame to the intermediate buffer dimension.

15:8

Reserved

Format:	MBZ

7:0

PictureWidthInMBsMinus1 (Picture Width Minus 1 in Macroblocks)

Format:	U8-1

This field indicates the width of the picture in unit of macroblocks. For example, for a 1920x1080 frame picture, PictureWidthInMBs equals 120 (1920 divided by 16). This field is used in VLD and IT modes

Value	Name	Description
[0,255]		[1,256] MB

2

31:24

Bitplane Buffer Pitch Minus 1

Format:	U7-1 Pitch in (Bytes - 1).

Specifies the bitplane buffer pitch in (#Bytes - 1). Bitplane buffer is a linear buffer. It is needed only when the bitplane is not encoded as raw, and therefore is present in the header explicitly. In VC1 Long Format, it is written by an application and later read by the HW. But in VC1 Short Format, it is written and read by H/W only. This field is specified for better performance

Value	Name
[0,FFFFFFFFh]	

Programming Notes
The pitch must be equal to PictureWidthInMBs/2.For VC1 Long Format: The pitch must be equal to PictureWidthInMBs/2.For VC1 Short Format: If Pic Width is less than or equal to 2K pixels, bitplane pitch is set to 64 (one cacheline; programmed as 63) bytes per MB row. If Pic Width is greater than 2K pixels, bitplane pitch is set to 128 (two cachelines; programmed as 127) bytes per MB row. This field is not used in IT mode, used in VLD mode only. For VC1 DXVA2 Short Format, the bitplane specification is between H/W and Driver only. For Long Format, application is responsible for allocation with the driver.

23:16

Reserved

Format:	MBZ

15

DmvSurfaceValid

Indicated when the DMV read surface is valid. This surface stored the direct motion vectors and Mb type. This field is set for B pictures that can refer to a previous P picture for DMV. If there is an I-picture before a B (in decoding order) then this field is not set (as a result, zero's DMV's will be assumed while decoding the B picture. That is, there is no explicit DMV buffer for an I-picture). Whne the current picture being decoded is an I, P or BI, this bit is set to 0, since there is no DMV read in these picture decoding process. This field is not used in IT mode, used in VLD

MFD_VC1_LONG_PIC_STATE

mode only.

14 **ImplicitQuantizer**

Derived by driver from QUANTIZER. This field is used in this implementation's VC1 VLD Long Format only, not used in IT and DXVA2 VC1.This bit is set to 1 when syntax element QUANTIZER=0, else its set to 0

13 **Interpolation Rounder Contro**

Used only in MC operation. This field specifies the rounding control value used in interpolation operation of motion prediction process. This field is used in VLD and IT modes.

Programming Notes
This bit field is taken from bRcontrol in DXVA_PictureParameters data structure

12 **SyncMarker**

Indicates whether sync markers are enabled/disabled. If enable, sync markers "may be" present in the current video sequence being decoded. It is a sequence level syntax element and is valid only for Simple and Main Profiles.

Value	Name	Description
0h	Not Present	Sync Marker is not present in the bitstream
1h	Maybe present	Sync Marker maybe present in the bitstream

Programming Notes
This field is only valid in VLD mode. For Simple Profile, SyncMarker must set to 0.For Main Profile, SyncMarker can be set to 0 or 1.This field is used in both this implementation's and MS DXVA2 VLD interface, but not used in IT mode.

11:8 **Motion Vector Mode**

This field indicates one of the following motion compensation interpolation modes for P and B pictures. The MC interpolation modes apply to prediction values of luminance blocks and are always in quarter-sample. For chrominance blocks, it always performs bilinear interpolation with either half-pel or quarter-pel precision. Before the polarity of Chroma Half-pel or Q-pel is reversed from DXVA2 Spec, now I have fixed it to match with DXVA2 VC1 Spec.

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

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

7 **RangeReductionScale**

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

Value	Name	Description

MFD_VC1_LONG_PIC_STATE

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

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

6

RangeReduction Enable

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

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

Programming Notes
This field is for Main Profile only. Simple Profile is always disable, and not applicable to Advanced Profile. This field is used in both VLD and IT modes. This is derived by driver from the history of RANGERED and RANGEREDFRM syntax elements (i.e. of forward/preceding reference picture) and those of the current picture. RANGERED is the same as (bPicOverflowBlocks >> 3) & 1. RANGEREDFRM is the same as (bPicDeblocked >> 5) & 1.For the current picture is a B picture, this field represents the state of the forward/preceding reference picture onlyDriver is responsible to keep RangeReductionScale, RangeReduction Enable and RANGERED Present Flag of current picture coherent.

5

LOOPFILTER Enable Flag

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

Value	Name	Description
0h	Disable	Disables loop filter
1h	Enable	Enables loop filter

4

Overlap Smoothing Enable Flag

This field is the decoded syntax element OVERLAP in bitstreamIndicates if Overlap smoothing is

MFD_VC1_LONG_PIC_STATE

ON at the picture levelThis field is used in both VLD and IT modes.

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

3 **Secondfield**

This flag is set for the second field in field pictures. This field is used in both VLD and IT modes.

2:1 **Reserved**

Format:	MBZ

0 **VC1 Profile**

specifies the bitstream profile. This field is used in both VLD and IT modes.

Value	Name	Description
0h	Disable	current picture is in Simple or Main Profile (No need to distinguish Simple and Main Profile)
1h	Enable	current picture is in Advanced Profile

Programming Notes
This is required because 128 is added for intra blocks post inverse transform in advanced profile and also to find out if Motion vectors are adjusted or not.

3 | 31 **Reserved**

Format:	MBZ

30:29 **CondOver**

This field is the decoded syntax element CONDOVER in a bitstream of advanced profile. It controls the overlap smoothing filter operation for an I frame or an BI frame when the picture level qualization step size PQUANT is 8 or lower. This field is used in this implementation's 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 this implementation's 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 this implementation's 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

MFD_VC1_LONG_PIC_STATE

<table>
<tr><td></td><td></td><td>Main ProfilesThis field is unique to this implementation's 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 this implementation's HW VLD Long Format interface.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>00b</td><td>Disable</td><td>Progressive Frame Picture</td></tr>
<tr><td>01b</td><td>Enable</td><td>Interlaced Frame Picture</td></tr>
<tr><td>10b</td><td></td><td>Field Picture with Top Field First</td></tr>
<tr><td>11b</td><td></td><td>Field Picture with Bottom Field First</td></tr>
</table></td></tr>
<tr><td></td><td>23:21</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>20:16</td><td><b>AltPQuant (Alternative Picture Quantization Value)</b>
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 this implementation's 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><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>12:8</td><td><b>PQuant (Picture Quantization Value)</b>
<table><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 this implementation's and DXVA2 VLD modes).</td></tr>
<tr><td></td><td>7:0</td><td><b>BScaleFactor</b>
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 this implementation's 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><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>29:28</td><td><b>UnifiedMvMode (Unified Motion Vector Mode)</b>
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</td></tr>
</table>

MFD_VC1_LONG_PIC_STATE

bicubic or bilinear Luma MC interpolation mode is duplicate information from Motion Vector Mode field, and is ignored here. This field is used in this implementation's VC1 VLD Long Format mode only, it is not used in DXVA2 VC1 VLD and IT modes.

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

27

FourMvSwitch (Four Motion Vector Switch)

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

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

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 this implementation's VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.

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

24

NumRef (Number of References)

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

Value	Name	Description

MFD_VC1_LONG_PIC_STATE

0h		One field referenced
1h		Two fields referenced

23:20 **BwdRefDist (Reference Distance)**

This field is valid only in B field pictures giving the value of BRFD. The field is ignored in P Picture. This field is unique to this implementation's 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 this implementation's VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.

Value	Name
[0, 15]	

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 this implementation's VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.

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

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

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

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

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

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

MFD_VC1_LONG_PIC_STATE

<table>
<tr><td></td><td></td><td>macroblocksBit 3: Bottom picture edge macroblocksThis field is unique to this implementation's VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.</td></tr>
<tr><td></td><td>3:2</td><td><b>AltPQuantConfig (Alternative Picture Quantization Configuration)</b>
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 this implementation's 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>00b</td><td></td><td>AltPQuant not used</td></tr>
<tr><td>01b</td><td></td><td>AltPQuant is used and applied to edge macroblocks only</td></tr>
<tr><td>10b</td><td></td><td>MQUANT is encoded in macroblock layer</td></tr>
<tr><td>11b</td><td></td><td>AltPQuant and PQuant are selected on macroblock basis</td></tr>
</table></td></tr>
<tr><td></td><td>1</td><td><b>HalfQP</b>
This field is used for inverse quantization of AC coefficients. It is valid only when PQuant is used. This field is unique to this implementation's VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.</td></tr>
<tr><td></td><td>0</td><td><b>PQuantUniform</b>
 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 this implementation's 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></td><td>Non-uniform</td></tr>
<tr><td>1h</td><td></td><td>Uniform</td></tr>
</table></td></tr>
<tr><td>5</td><td>31</td><td><b>BitplanePresentFlag (Bitplane Buffer Present Flag)</b>
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 this implementation's 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></td><td>bitplane buffer is not present</td></tr>
<tr><td>1h</td><td></td><td>bitplane buffer is present</td></tr>
</table></td></tr>
<tr><td></td><td>30</td><td><b>ForwardMbRaw</b>
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 this implementation's 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>
</table></td></tr>
</table>

MFD_VC1_LONG_PIC_STATE

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 this implementation's 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 this implementation's 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 this implementation's 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 this implementation's 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 this implementation's VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.

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

24 **FieldTxRaw**

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

MFD_VC1_LONG_PIC_STATE

<table>
<tr><td></td><td></td><td>
<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>
</td></tr>
<tr><td></td><td>23</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table>
</td></tr>
<tr><td></td><td>22:20</td><td><b>MvTab (Motion Vector Table)</b>
<table><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 this implementation's VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.
</td></tr>
<tr><td></td><td>19:18</td><td><b>FourMvBpTab (4-MV Block Pattern Table)</b>
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 this implementation's VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.
</td></tr>
<tr><td></td><td>17:16</td><td><b>TwoMvBpTab (2MV Block Pattern Table)</b>
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 this implementation's VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.
</td></tr>
<tr><td></td><td>15:14</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table>
</td></tr>
<tr><td></td><td>13:12</td><td><b>TransType (Picture-level Transform Type)</b>
<table><tr><td>Format:</td><td>U2</td></tr></table>
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
</td></tr>
</table>

MFD_VC1_LONG_PIC_STATE

<table>
<tr><td></td><td></td><td>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 this implementation's VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.</td></tr>
<tr><td></td><td>11</td><td><b>TransTypeMbFlag (Macroblock Transform Type Flag)</b>
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 this implementation's 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></td><td>variable transform type in macroblock layer</td></tr>
<tr><td>1h</td><td></td><td>use picture level transform type TransType</td></tr>
</table></td></tr>
<tr><td></td><td>10:8</td><td><b>MbModeTab (Macroblock Mode Table)</b>
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 this implementation's VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.</td></tr>
<tr><td></td><td>7:6</td><td><b>TransAcY (Picture-level Transform Luma AC Coding Set Index, TRANSACTABLE2</b>
BitFieldDesc</td></tr>
<tr><td></td><td>5:4</td><td><b>TransAcUV (Picture-level Transform Chroma AC Coding Set Index, TRANSACTABLE)</b>
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 this implementation's VC1 VLD Long format mode, and is not used in IT and DXVA2 VC1 modes.</td></tr>
<tr><td></td><td>3</td><td><b>TransDcTab (Intra Transform DC Table)</b>
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 this implementation's 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></td><td>The high motion tables</td></tr>
<tr><td>1h</td><td></td><td>The low motion tables</td></tr>
</table></td></tr>
</table>

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

MFD_VC1_SHORT_PIC_STATE

Source: VideoCS
Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFD_VC1_SHORT_PIC_STATE Format: OpCode
	26:24	**Media Command Opcode** Default Value: 2h VC1_DEC Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h Format: OpCode
	20:16	**SubOpcode B** Default Value: 0h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 0003h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:24	**Reserved** Format: MBZ
	23:16	**Picture Height** Format: U8-1 Picture Height in Macroblocks This field indicates the height of the picture in unit of macroblocks. For example, for a 1920x1080 frame picture, PictureHeightInMBs equals 68 (1080 divided by 16, and rounded up, i.e. effectively specified as 1088 instead). This field is used in VLD and IT modes. Note: Even though the Advanced Profile allows frame dimensions (width, height) to not be aligned to macroblock boudary, it doesn't affect the bitstream decoding. And it is preferable to use 'intermediate buffer' that is macroblock aligned for decoding. In order to simplify the out-of-bound reference pixel access, the out-of-bound extrapolation rule in VC1 spec can be used to expand the expected decoded frame to the intermediate buffer dimension. Value | Name | Description [0,255] | | [1, 256] MB
	15:8	**Reserved** Format: MBZ

<table>
<tr><th colspan="3">MFD_VC1_SHORT_PIC_STATE</th></tr>
<tr><td></td><td>7:0</td><td><b>Picture Width</b>
<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></tr><tr><td>[0,255]</td><td></td><td>[1, 256] MB</td></tr></table></td></tr>
<tr><td>2</td><td>31:24</td><td><b>Bitplane Buffer Pitch Minus 1</b>
<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, it is written by an application and later read by the HW. In VC1 Long Format, it is written by an application, and later read by the HW. But in VC1 Short Format, it is written and read by H/W only. This field is specified for better performance. For VC1 Long Format: The pitch must be equal to PictureWidthInMBs/2. For 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><b>Interpolation Rounder Control</b>
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><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>19:16</td><td><b>Motion Vector Mode</b>
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.</td></tr>
<tr><td></td><td>15</td><td><b>DmvSurfaceValid</b>
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.</td></tr>
</table>

MFD_VC1_SHORT_PIC_STATE

<table>
<tr><td></td><td>14:12</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11</td><td><b>VC1 Profile</b>
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.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td><b>[Default]</b></td><td>current picture is in Simple or Main Profile (No need to distinguish Simple and Main Profile)</td></tr><tr><td>1h</td><td></td><td>current picture is in Advanced Profile</td></tr></table></td></tr>
<tr><td></td><td>10:6</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>5</td><td><b>Backward Prediction Present Flag</b>
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.</td></tr>
<tr><td></td><td>4</td><td><b>Intra Picture Flag</b>
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.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td></td><td>entire picture can have a mixture of intra and inter MB type or just inter MB type.</td></tr><tr><td>1h</td><td></td><td>entire picture is coded in intra MB type</td></tr></table></td></tr>
<tr><td></td><td>3</td><td><b>SecondField</b>
This flag is set for the second field in field pictures. This field is used in both VLD and IT modes.</td></tr>
<tr><td></td><td>2</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>1:0</td><td><b>Picture Structure</b>
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.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>01b</td><td></td><td>top field (bit 0)</td></tr></table></td></tr>
</table>

<table>
<tr><th colspan="3">MFD_VC1_SHORT_PIC_STATE</th></tr>
<tr><td></td><td></td><td><table>
<tr><td>10b</td><td></td><td>bottom field (bit 1)</td></tr>
<tr><td>11b</td><td></td><td>frame (both fields are present)</td></tr>
<tr><td>00b</td><td></td><td>illegal</td></tr>
</table></td></tr>
<tr><td>3</td><td>31</td><td><b>Reserved</b><table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>30</td><td><b>Overlap Smoothing Enable Flag</b>
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<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Disable</td><td>to disable overlap smoothing filter</td></tr>
<tr><td>1h</td><td>Enable</td><td>to enable overlap smoothing filter</td></tr>
</table></td></tr>
<tr><td></td><td>29</td><td><b>Range Reduction Scale</b><table>
<tr><td>Access:</td><td>None</td></tr>
</table>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.<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Disable <b>[Default]</b></td><td>Scale down reference picture by factor of 2</td></tr>
<tr><td>1h</td><td>Enable</td><td>Scale up reference picture by factor of 2</td></tr>
</table></td></tr>
<tr><td></td><td>28</td><td><b>Range Reduction Enable</b>
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.<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Disable <b>[Default]</b></td><td>Range reduction is not performed</td></tr>
<tr><td>1h</td><td>Enable</td><td>Range reduction is performed</td></tr>
</table></td></tr>
<tr><td></td><td>27:24</td><td><b>Reserved</b></td></tr>
</table>

MFD_VC1_SHORT_PIC_STATE

Format:	MBZ

23:22 **Progressive Pic Type**

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

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

21 **Reserved**

Format:	MBZ

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

Access:	None

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

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

15:14 **QUANTIZER**

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

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

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

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

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

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

MFD_VC1_SHORT_PIC_STATE

<table>
<tr><td></td><td></td><td>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></td></tr>
<tr><td></td><td>10:8</td><td><b>MAXBFRAMES</b>
Number of consecutive B Frames.</td></tr>
<tr><td></td><td>7</td><td><b>PANSCAN Present Flag</b>
<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></td></tr>
<tr><td></td><td>6</td><td><b>REFDIST_FLAG</b>
For header parsing REFDIST. This is used in DXVA2 VC1 VLD mode only, not used in IT and our VC1 VLD modes.</td></tr>
<tr><td></td><td>5</td><td><b>LOOPFILTER Enable Flag</b>
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.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td></td><td>In-Loop-Deblocking-Filter is disabled</td></tr>
<tr><td>1</td><td></td><td>In-Loop-Deblocking-Filter is enabled</td></tr>
</table></td></tr>
<tr><td></td><td>4</td><td><b>FastUVMCFlag (Fast UV Motion Compensation Flag)</b>
 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.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td></td><td>no rounding</td></tr>
<tr><td>1h</td><td></td><td>quarter-pel offsets to half/full pel positions</td></tr>
</table></td></tr>
<tr><td></td><td>3</td><td><b>EXTENDED_MV Present Flag</b>
BitFieldDesc
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td></td><td>Extended_MV is not present in the picture header</td></tr>
<tr><td>1h</td><td></td><td>Extended_MV is present in the picture header</td></tr>
</table></td></tr>
<tr><td></td><td>2:1</td><td><b>DQUANT</b></td></tr>
</table>

<table>
<tr><th colspan="3">MFD_VC1_SHORT_PIC_STATE</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td>Access:</td><td>None</td></tr>
<tr><td>Format:</td><td>U2</td></tr>
</table>
Use for Picture Header Parsing of VOPDUANT elements
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>[Default]</td><td></td></tr>
<tr><td>00b</td><td></td><td>no VOPDQUANT elements; Quantizer cannot vary in frame, same quantization step size PQUANT is used for all MBs in the frame</td></tr>
<tr><td>01b</td><td></td><td>refer to VC1 Spec. for all the MB position dependent quantizer selection</td></tr>
<tr><td>10b</td><td></td><td>The macroblocks located on the picture edge boundary shall be quantized with ALTPQUANT while the rest of the macroblocks shall be quantized with PQUANT.</td></tr>
<tr><td>11b</td><td>Reserved</td><td></td></tr>
</table>
</td></tr>
<tr><td></td><td>0</td><td>
<b>VSTRANSFORM flag</b>
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Disable</td><td>variable-sized transform coding is not enabled</td></tr>
<tr><td>1h</td><td>Enable</td><td>variable-sized transform coding is enabled</td></tr>
</table>
</td></tr>
<tr><td>4</td><td>31:29</td><td>
<b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ (for possible future change to BFraction Enumeration)</td></tr>
</table>
</td></tr>
<tr><td></td><td>28:24</td><td>
<b>BFraction Enumeration</b>
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 our 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)
</td></tr>
<tr><td></td><td>23</td><td>
<b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ Advanced Profile only; RANGE_MAPY_FLAG Range Mapping not supported</td></tr>
</table>
</td></tr>
<tr><td></td><td>22:20</td><td>
<b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ Advanced Profile only; RANGE_MAPY Range Mapping not supported</td></tr>
</table>
</td></tr>
<tr><td></td><td>19</td><td>
<b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ Advanced Profile only; RANGE_MAPUV_FLAG Range Mapping not supported</td></tr>
</table>
</td></tr>
<tr><td></td><td>18:16</td><td><b>Reserved</b></td></tr>
</table>

MFD_VC1_SHORT_PIC_STATE

<table>
<tr><td></td><td></td><td><table><tr><td>Format:</td><td>MBZ Advanced Profile only; RANGE_MAPUV Range Mapping not supported</td></tr></table></td></tr>
<tr><td></td><td>15:9</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>8</td><td><b>4MV Allowed Flag</b></td></tr>
<tr><td></td><td>7</td><td><b>POSTPROC Flag</b></td></tr>
<tr><td></td><td>6</td><td><b>PULLDOWN</b></td></tr>
<tr><td></td><td>5</td><td><b>INTERLACE</b></td></tr>
<tr><td></td><td>4</td><td><b>TFCNTRFLAG</b></td></tr>
<tr><td></td><td>3</td><td><b>FINTERFLAG</b></td></tr>
<tr><td></td><td>2</td><td><b>REFPIC Flag</b>
For a BI picture, REFPIC flag must set to 0For I and P picture, REFPIC flag must set to 0.For a B picture, REFPIC flag must set to 0, except for a B-field in interlaced field mode which can be 0 or 1 (e.g. the top B field can be used as a reference for decoding its corresponding bottom B-field in a field pair). In VLD mode, this flag cannot be used as an optimization signaling for an I or P picture that is not used as a reference picture. This field is used in both DXVA2 VC1 VLD mode and IT mode. It is the same parameter as bPicDeblockConfined[bit2] in DXVA2 VC1 spec.The Intra Picture Flag, Backward Prediction Present Flag and RefPicFlag are used to derive the picture type, as specified in PTYPE for a frame, and in FPTYPE for a field, in DXVA2 VC1 VLD and IT mode.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td></td><td>the current picture after decoded, will never used as a reference picture</td></tr><tr><td>1h</td><td></td><td>the current picture after decoded, will be used as a reference picture later</td></tr></table></td></tr>
<tr><td></td><td>1</td><td><b>PSF</b></td></tr>
<tr><td></td><td>0</td><td><b>EXTENDED_DMV Present Flag</b><table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td><b>[Default]</b></td><td>Extended_DMV is not present in the picture header</td></tr><tr><td>1h</td><td></td><td>Extended_DMV is present in the picture header</td></tr></table></td></tr>
</table>

MFX_AVC_DIRECTMODE_STATE

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 Format: OpCode
	23:21	**SubOpcodeA** Default Value: 0h MEDIA_ Format: OpCode
	20:16	**SubOpcodeB** Default Value: 2h Desc Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 0043h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:6	**Direct MV Buffer Base Address for Picture 0 (current or reference top field)** Format: GraphicsAddress[31:6] This field provides the base address of the DMV write buffer to store motion vectors decoded in the current picture (top field), which may be used later as a collocated motion information read buffer of the associated reference picture in decoding subsequent B-pictures that have MB coded in direct mode. It is a private buffer used by the MPR hardware only. Its content is not accessed by software. This buffer must be 64-byte cacheline aligned. The write buffer size is 557,056 bytes for 1 frame. Scalable with frame height, but do not scale with frame width as the hardware assumes frame width (in MBs) fixed at 128 (smallest power of 2 value larger than 120 - 1920x1088 screen resolution)It is only valid if the current picture is a progressive frame, MbAff frame, or a top field. There are a total of 32 reference picture (previously decoded) Direct MV Buffers (0 to 31, not including the DMV write buffer 32 and 33 of the current picture) to read in

MFX_AVC_DIRECTMODE_STATE

the corresponding collocated DMV and motion information. For reference picture, these 32 DMV read Buffers can be indexed by the frame_store_ID[4:0], which is obtained from RefPicList L0/L1[RefPicIdx]. frame_Store_IDbit[0] (indicator for Top/Bottiom Field). For writing out motion information during the decoding of the current picture, all 34 DMV buffers can be addressed by [img_dec_fs_idc[4:0]<<1 + img_structure[1]].

5:4 **Direct MV Buffer - Arbitration Priority Control**

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

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

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

3 **Reserved**

2 **Direct MV Buffer - Graphics Data Type (GFDT) for Picture 0**

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

Value	Name	Description
0h	Disable	Desc
1h	Enable	Desc

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

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

This field controls cacheability.

Value	Name	Description
00b	Use cacheability control bits from GTT entry	Desc
01b	Data is not cached	Desc
11b	Data is cached	

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

MFX_AVC_DIRECTMODE_STATE

2	31:6	**Direct MV Buffer Base Address for Picture 1 (current or reference bottom field)** Format: GraphicsAddress[31:6] This field provides the base address of the DMV read/write buffer for the current or reference picture (bottom field). It is paired with the DMV Buffer of Picture 0 for MB pair retrieval during read. It follows the same format specification as DMV buffer for Picture 0It is only valid if the current picture is a bottom field. It is also valid
	5:4	**Direct MV Buffer - Arbitration Priority Con** Format: U2 This field is ignored in H/W, and assumes the same value as of Picture 0 DMV Buffer specification bit[5:4] above.
	3	**Reserved**
	2	**Direct MV Buffer - Graphics Data Type (GFDT) for Picture 1** Format: U1 This field is ignored in H/W, and assumes the same value as of Picture 0 DMV Buffer specification bit[2] above.
	1:0	**Direct MV Buffer -Cacheability Control for Picture 1** Format: U2 This field is ignored in H/W, and assumes the same value as of Picture 0 DMV Buffer specification bit[1:0] above.
3..32	31:6	**Direct MV Buffer Base Address for Reference Frame 2 to 31** Format: GraphicsAddress[31:6] This field provides the base address of the DMV buffer for reference frame 2 to 31. They are needed if the current B-Picture has MBs coded in direct mode. It is a private buffer used by the MPR hardware only. Its content is not accessed by software. All these buffers must be 64-byte cacheline aligned. There are a total of 32 possible Direct MV Read Buffers (not including the current write buffer of the current picture) to read in the corresponding DMV. Each read buffer size is 557,056 bytes for 1 frame (the selected colPic). Scalable with frame height, but do not scale with frame width as the hardware assumes frame width (in MBs) fixed at 128 (smallest power of 2 value larger than 120 - 1920x1088 screen resolution). The adjacent DMV buffers are paired ([2 and 3], [4 and 5], [N and N+1], ..[30 and 31]).
	5:4	**Direct MV Buffer - Arbitration Priority Control** Format: U2 This field is ignored in H/W, and assumes the same value as of Picture 0 DMV Buffer specification bit[5:4] above.
	3	**Reserved**
	2	**Direct MV Buffer - Graphics Data Type (GFDT) for Reference Frame 2 to 31** Format: U1

MFX_AVC_DIRECTMODE_STATE

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

<table>
<tr><th colspan="3">MFX_AVC_DIRECTMODE_STATE</th></tr>
<tr><td></td><td></td><td>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.</td></tr>
</table>

MFX_AVC_IMG_STATE

Source:	VideoCS
Length Bias:	2

This must be the very first command to issue after the surface state, the pipe select and base address setting commands

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 The value for FrameSizeInMBs must match the product of FrameWidthInMBs and FrameHeightInMBs. Max. Screen resolution is therefore limited to 256 x 256 in MB unit. This parameter is specified for Intel interface only. Value | Name | Description [0,16383] | | representing Number of MBs [1,16384]
2	31:24	**Reserved**

<table>
<tr><th colspan="3">MFX_AVC_IMG_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>23:16</td><td><b>Frame Height</b><table><tr><td>Format:</td><td>U8-1 in MB unit</td></tr></table>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.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>[0,255]</td><td></td><td>representing height [1,256]</td></tr></table></td></tr>
<tr><td></td><td>15:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>Frame Width</b><table><tr><td>Format:</td><td>U8-1 in MB unit</td></tr></table>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.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>[0,255]</td><td></td><td>representing width [1,256]</td></tr></table></td></tr>
<tr><td>3</td><td>31:29</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>28:24</td><td><b>Second Chroma QP Offset</b>
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</td></tr>
<tr><td></td><td>23:21</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>20:16</td><td><b>First Chroma QP Offset</b>
Signed integer value. It should be in the range of -12 to +12 (according to AVC spec). It</td></tr>
</table>

MFX_AVC_IMG_STATE

<table>
<tr><td></td><td></td><td>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</td></tr>
<tr><td></td><td>15:14</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table>
</td></tr>
<tr><td></td><td>13</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table>
</td></tr>
<tr><td></td><td>12</td><td><b>Weighted_Pred_Flag</b>
<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 <b>[Default]</b></td><td>specifies that weighted prediction is not used for P and SP slices</td></tr>
<tr><td>1</td><td>Enable</td><td>specifies that weighted prediction is used for P and SP slices</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>This field must set to '0' for B and I pictures.</td></tr>
</table>
</td></tr>
<tr><td></td><td>11:10</td><td><b>Weighted_BiPred_Idc</b>
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>DEFAULT <b>[Default]</b></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>
</td></tr>
<tr><td></td><td>9:8</td><td><b>ImgStruct - Image Structure, img_structure[1:0]</b>
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</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th colspan="3">MFX_AVC_IMG_STATE</th></tr>
<tr><td></td><td></td><td><table><tr><td>parameter is specified for this interface only.</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table>Used to be 8-bit img_dec_fs</td></tr>
<tr><td>4</td><td>31:16</td><td><b>MinFrameWSize</b><table><tr><td>Default Value:</td><td>0h</td></tr><tr><td>Format:</td><td>U16</td></tr></table><b>Minimum Frame Size [15:0] (in Word, 16-bit)(Encoder Only)</b> Mininum Frame Size is specified to compensate for 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. 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 <b>FrameBitRateMax (DWORD 10 bits</b> 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.</td></tr>
<tr><td></td><td>15</td><td><b>MbStatEnabled</b><table><tr><td>Format:</td><td>Enable</td></tr></table>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.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td>Disable</td><td>Disable Reading of Macroblock Status Buffer</td></tr><tr><td>1</td><td>Enable</td><td>Enable Reading of Macroblock Status Buffer</td></tr></table></td></tr>
<tr><td></td><td>14</td><td><b>LoadSlicePointerFlag</b><table><tr><td>Format:</td><td>Enable</td></tr></table>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.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td>Disable</td><td>Load BitStream Pointer only once for the first slice of a frame</td></tr><tr><td>1</td><td>Enable</td><td>Load/reload BitStream Pointer only once for the each slice, reload the</td></tr></table></td></tr>
</table>

MFX_AVC_IMG_STATE

		start location of the bitstream buffer from the Indirect PAK-BSE Object Data Start Address field

13 **Reserved**

12 **MvUnpackedFlag**

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

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

11:10 **ChromaFormatIdc**

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

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

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

9 **Reserved**

Format:	MBZ

8 **MbMvFormatFlag**

Use MB level MvFormat flag (Encoder Only)

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
0	CAVLC bit-serial encoding mode

MFX_AVC_IMG_STATE

1	CABAC bit-serial encoding mode.

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.

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.

MFX_AVC_IMG_STATE

<table>
<tr><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>4x4</td><td>no 8x8 IDCT Transform, only 4x4 IDCT transform blocks are present</td></tr>
<tr><td>1</td><td>8x8</td><td>8x8 Transform is allowed</td></tr>
</table>
</td></tr>
<tr><td></td><td>2</td><td>
<b>FrameMbOnlyFlag</b>
Frame MB only flag, frame_mbs_only_flagIt is set to the value of the syntax element in the current active SPS.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>FALSE</td><td>not true ; effectively enables the possibility of MBAFF mode.</td></tr>
<tr><td>1</td><td>TRUE</td><td>true, only frame MBs can occur in this sequence, hence disallows the MBAFF mode and field picture.</td></tr>
</table>
</td></tr>
<tr><td></td><td>1</td><td>
<b>MbaffFlameFlag</b>
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.
<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>
</td></tr>
<tr><td></td><td>0</td><td>
<b>FieldPicFlag</b>
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>
</td></tr>
<tr><td>5
[ExistsIf]Encode Only</td><td>31</td><td>
<b>Trellis Quantization Enabled (TQEnb)</b>
<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></tr>
<tr><td>0h</td><td>Disable</td><td>Use Normal</td></tr>
</table>
</td></tr>
<tr><td></td><td>30:28</td><td>
<b>Trellis Quantization Rounding (TQR)</b>
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.
</td></tr>
</table>

MFX_AVC_IMG_STATE

<table>
<tr><td></td><td>27</td><td><b>Trellis Quantization Chroma Disable (TQChromaDisable)</b>
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.</td></tr>
<tr><td></td><td>26:21</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>20:17</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>16</td><td><b>NonFirstPassFlag</b>
This signals the current pass is not the first pass. It will imply designate HW behavior: e.g<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Disable</td><td>Always use the MbQpY from initial PAK inline object for all passes of PAK</td></tr><tr><td>1h</td><td>Enable</td><td>Use MbQpY from stream-out buffer if MbRateCtrlFlag is set to 1</td></tr></table></td></tr>
<tr><td></td><td>15:13</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>12</td><td><b>InterMbZeroCbpFlag - InterMB Force CBP to Zero Control Flag</b>
Inter MB Force CBP ZERO mask.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Disable</td><td>No Effect</td></tr><tr><td>1h</td><td>Enable</td><td>Zero out all A/C coefficients for the inter MB violating Inter Confirmance</td></tr></table></td></tr>
<tr><td></td><td>11:10</td><td><b>MinFrameWSizeUnits</b>
This field is the Minimum Frame Size Units<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>00b</td><td>compatibility mode</td><td>Minimum Frame Size is in old mode (words, 2bytes)</td></tr><tr><td>01b</td><td>16 byte</td><td>Minimum Frame Size is in 16bytes</td></tr><tr><td>10b</td><td>4Kb</td><td>Minimum Frame Size is in 4Kbytes</td></tr><tr><td>11b</td><td>16Kb</td><td>Minimum Frame Size is in 16Kbytes</td></tr></table></td></tr>
<tr><td></td><td>9</td><td><b>MbRateCtrlFlag - MB level Rate Control Enabling Flag</b>
MB Rate Control conformance mask<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Disable</td><td>Apply accumulative delta QP for consecutive passes on top of the macroblock QP values in inline data</td></tr><tr><td>1h</td><td>Enable</td><td>Apply RC QP delta to suggested QP values in Macroblock Status Buffer except the first pass.</td></tr></table><table><tr><th>Programming Notes</th></tr><tr><td>This field is ignored when MacroblockStatEnable is disabled or MB level Rate control</td></tr></table></td></tr>
</table>

MFX_AVC_IMG_STATE

<table>
<tr><td></td><td></td><td><table><tr><td>flag for the current MB is disable in Macroblock Status Buffer.</td></tr></table></td></tr>
<tr><td></td><td>8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7</td><td><b>Intra/InterMbIpcmFlag - ForceIPCMControlMask</b>
This field is to Force <b>IPCM</b> for Intra or Inter Macroblock size conformance mask.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Disable</td><td>Do not change intra macroblocks even.</td></tr><tr><td>1h</td><td>Enable</td><td>Change intra macroblocks MB_type to IPCM.</td></tr></table><table><tr><th>Programming Notes</th></tr><tr><td>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.</td></tr></table></td></tr>
<tr><td></td><td>6:4</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>3</td><td><b>FrameSzUnderFlag - FrameBitRateMinReportMask</b>
This is a mask bit controlling if the condition of frame level bit count is less than FrameBitRateMin<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Disable</td><td>Do not update bit0 of MFC_IMAGE_STATUS control register.</td></tr><tr><td>1h</td><td>Enable</td><td>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.</td></tr></table></td></tr>
<tr><td></td><td>2</td><td><b>FrameSzOverFlag - FrameBitRateMaxReportMask</b>
This is a mask bit controlling if the condition of frame level bit count exceeds FrameBitRateMax.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td>Disable</td><td>Do not update bit0 of MFC_IMAGE_STATUS control register.</td></tr><tr><td>1</td><td>Enable</td><td>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.</td></tr></table></td></tr>
<tr><td></td><td>1</td><td><b>InterMbMaxBitFlag - InterMBMaxSizeReportMask</b>
This is a mask bit controlling if the condition of any inter MB in the frame exceeds InterMBMaxSize.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td>Disable</td><td>Do not update bit0 of MFC_IMAGE_STATUS control register.</td></tr><tr><td>1</td><td>Enable</td><td>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.</td></tr></table></td></tr>
<tr><td></td><td>0</td><td><b>IntraMbMaxBitFlag - IntraMBMaxSizeReportMask</b>
This is a mask bit controlling if the condition of any intra MB in the frame exceeds IntraMBMaxSize.</td></tr>
</table>

MFX_AVC_IMG_STATE

<table>
<tr><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Disable</td><td>Do not update bit0 of MFC_IMAGE_STATUS control register.</td></tr>
<tr><td>1</td><td>Enable</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>6
[ExistsIf]Encode Only</td><td>31:28</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>27:16</td><td><b>InterMbMaxSz</b>
<table><tr><td>Format:</td><td>U12</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></td><td>15:12</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11:0</td><td><b>IntraMbMaxSz</b>
<table><tr><td>Exists If:</td><td>//Intra Only</td></tr><tr><td>Format:</td><td>U12</td></tr></table>
<table><tr><td>This field, Intra MB Conformance Max size limit,indicates the allowed max bit count size for Intra MB</td></tr><tr><td>All IPCM MBs should ignore this Max size limit.</td></tr></table></td></tr>
<tr><td>7
[ExistsIf]Encode Only</td><td>31:1</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>0</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>8
[ExistsIf]Encode Only</td><td>31:24</td><td><b>SliceDeltaQpMax[3]</b>
<table><tr><td>Format:</td><td>S7</td></tr></table>
<table><tr><td>Range: [0:MAX_QP_DELTA]</td></tr><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></table></td></tr>
<tr><td></td><td>23:16</td><td><b>SliceDeltaQpMax[2]</b>
<table><tr><td>Format:</td><td>U8</td></tr></table>
<table><tr><td>Range: [0:MAX_QP_DELTA]</td></tr><tr><td>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</td></tr></table></td></tr>
</table>

MFX_AVC_IMG_STATE

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

<table>
<tr><th colspan="3">MFX_AVC_IMG_STATE</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td>Range: [0:MAX_QP_DELTA]</td></tr>
<tr><td>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)).</td></tr>
</table>
</td></tr>
<tr><td></td><td>7:0</td><td><b>SliceDeltaQpMin[0]</b>
<table>
<tr><td>Format:</td><td>S7</td></tr>
</table>
<table>
<tr><td>Range: [0:MAX_QP_DELTA]</td></tr>
<tr><td>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).</td></tr>
</table>
</td></tr>
<tr><td>10
[ExistsIf]Encode Only</td><td>31</td><td><b>FrameBitrateMaxUnit</b>
This field is the Frame Bitrate Maximum Limit Units.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>Byte</td><td>FrameBitRateMax is in units of 32 Bytes when FrameBitrateMaxUnitMode is 1 and in units of 128 Bytes if FrameBitrateMaxUnitMode is 0</td></tr>
<tr><td>1</td><td>Kilo Byte</td><td>FrameBitRateMax is in units of 4KBytes Bytes when FrameBitrateMaxUnitMode is 1 and in units of 16KBytes if FrameBitrateMaxUnitMode is 0</td></tr>
</table>
</td></tr>
<tr><td></td><td>30</td><td><b>FrameBitrateMaxUnitMode</b>
This field is the Frame Bitrate Maximum Limit Units.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>compatibility mode</td><td>FrameBitRateMaxUnit is in old mode (128b/16Kb)</td></tr>
<tr><td>1h</td><td>New mode</td><td>FrameBitRateMaxUnit is in new mode (32byte/4Kb)</td></tr>
</table>
</td></tr>
<tr><td></td><td>29:16</td><td><b>FrameBitRateMax</b>
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..
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0-512KB</td><td></td><td>The programmable range is 0-512KB when FrameBitrateMaxUnit is 0.</td></tr>
<tr><td>0-8190KB</td><td></td><td>The programmable range is 0-8190KB when FrameBitrateMaxUnit is 1.</td></tr>
</table>
</td></tr>
<tr><td></td><td>15</td><td><b>FrameBitrateMinUnit</b></td></tr>
</table>

MFX_AVC_IMG_STATE

<table>
<tr><td></td><td></td><td>This field is the Frame Bitrate Minimum Limit Units.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>Byte</td><td>FrameBitRateMax is in units of 32 Bytes when FrameBitrateMinUnitMode is 1 and in units of 128 Bytes if FrameBitrateMinUnitMode is 0</td></tr>
<tr><td>1</td><td>Kilo Byte</td><td>FrameBitRateMax is in units of 4KBytes Bytes when FrameBitrateMaxUnitMode is 1 and in units of 16KBytes if FrameBitrateMaxUnitMode is 0</td></tr>
</table></td></tr>
<tr><td></td><td>14</td><td><b>FrameBitrateMinUnitMode</b>
This field is the Frame Bitrate Minimum Limit Units.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Compatibility mode</td><td>FrameBitRateMaxUnit is in old mode (128b/16Kb)</td></tr>
<tr><td>1h</td><td>New mode</td><td>FrameBitRateMaxUnit is in new mode (32byte/4Kb)</td></tr>
</table></td></tr>
<tr><td></td><td>13:0</td><td><b>FrameBitRateMin</b>
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.</td></tr>
<tr><td>11
[ExistsIf]Encode Only</td><td>31</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>30:16</td><td><b>FrameBitRateMaxDelta</b>
<table>
<tr><td>Format:</td><td>U15</td></tr>
</table>
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.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0-1024KB</td><td></td><td>The Programmable range 0-1024KB when FrameBitRateMaxUnit is 0.</td></tr>
<tr><td>0-16380KB</td><td></td><td>The Programmable range is 0-16380KB when FrameBitRateMaxUnit is 1.</td></tr>
<tr><td>0h</td><td><b>[Default]</b></td><td></td></tr>
</table></td></tr>
<tr><td></td><td>15</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>14:0</td><td><b>FrameBitRateMinDelta</b>
<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</td></tr>
</table></td></tr>
</table>

MFX_AVC_IMG_STATE

		12, 13 and 14 should be 0.Note: HW requires the following condition FrameBitRateMinDelta <= 2*FrameBitRateMinMust be true, otherwise it may cause unpredicted behavior.
12	31:21	**Reserved** Format: MBZ
	20	**Reserved** Format: MBZ
	19	**Reserved** Format: MBZ
	18:16	**Reserved** Format: MBZ
	15:0	**Reserved** Format: MBZ
13	31:30	**Reserved** Format: MBZ
	29	**Current Picture Has Performed MMCO5** Set to 1 if the current Pic has performed the memory_management_control_operation = = 5.
	28:24	**Number of Reference Frames** Format: U5 Range: Range 0 to MaxDpbSize (=16 for Level 4.1) Specifies the maximum number of reference frames (frames, field pairs, unpaired field) existed in the current DBP for decoding the current picture.
	23:22	**Reserved** Format: MBZ
	21:16	**Number of Active Reference Pictures from L1** Format: U6-1 Specifies the initial maximum reference index value minus 1 to access the L1 Reference List. It is extracted from PPS. It corresponds to the number of active reference pictures from L1 to decode the current picture. It can be modified by the slice header if num_ref_idx_active_override_flag is set. Only valid for B picture. **Value** — **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

<table>
<tr><th colspan="3">MFX_AVC_IMG_STATE</th></tr>
<tr><td></td><td></td><td>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.
<table><tr><th>Value</th><th>Name</th></tr><tr><td>[0,31]</td><td></td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>Initial QP Value</b>
<table><tr><td>Format:</td><td>S7</td></tr></table>
<table><tr><th>Description</th></tr><tr><td>Range: [-26,25]</td></tr><tr><td>Short Format Only</td></tr><tr><td>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.</td></tr></table></td></tr>
<tr><td>14
[ExistsIf] Short Format only</td><td>31:24</td><td><b>Log2_max_pic_order_cnt_lsb_minus4</b>
<table><tr><td>Exists If:</td><td>//Short Format Only</td></tr></table>
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</td></tr>
<tr><td></td><td>23:16</td><td><b>Log2_max_frame_num_minus4</b>
<table><tr><td>Exists If:</td><td>//Short Format Only</td></tr></table>
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.</td></tr>
<tr><td></td><td>15</td><td><b>deblocking_filter_control_present_flag</b>
<table><tr><td>Exists If:</td><td>//Short Format Only</td></tr></table>
It is a PPS syntax element, indicates if more deblocking filter control syntax elements are present in the slice header.</td></tr>
<tr><td></td><td>14:12</td><td><b>num_slice_groups_minus1</b>
<table><tr><td>Exists If:</td><td>//Short Format Only</td></tr></table>
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</td></tr>
<tr><td></td><td>11</td><td><b>redundant_pic_cnt_present_flag</b>
<table><tr><td>Exists If:</td><td>//Short Format Only</td></tr></table>
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.</td></tr>
<tr><td></td><td>10:8</td><td><b>slice_group_map_type</b>
<table><tr><td>Exists If:</td><td>//Short Format Only</td></tr></table></td></tr>
</table>

MFX_AVC_IMG_STATE

		It is a PPS syntax element.Use for Slice Header parsing only, to read in slice_group_change_cycle, if any, but is not used by H/W, i.e. no slice group support.
	7:4	**Reserved** Format: | MBZ IDR flag is decoded from NAL Header Byte
	3:2	**Pic_order_cnt_type** Exists If: | //Short Format Only It is a SPS syntax element.Use for Slice Header parsing only.
	1	**Delta_pic_order_always_zero_flag** Exists If: | //Short Format Only It is a SPS syntax element.Use for Slice Header parsing only.
	0	**Pic_order_present_flag** Exists If: | //Short Format Only It is a PPS syntax element.Use for Slice Header parsing only.
15 [ExistsIf] Short Format only	31:16	**Curr Pic Frame Num** Exists If: | //Short Format Only Format: | U16 Derived from Slice Header syntax element
	15:0	**Slice Group Change Rate** Exists If: | //Short Format Only Format: | U16-1 It is a PPS syntax element Use for Slice Header parsing only, to read in slice_group_change_cycle, if any, but is not used by H/W, i.e. no slice group support.
16 [ExistsIf]: Short Format only	31:0	**Reserved**

MFX_AVC_REF_IDX_STATE

Source:	VideoCS
Length Bias:	2

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

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

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

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

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

<table>
<tr><th colspan="3">MFX_AVC_REF_IDX_STATE</th></tr>
<tr><td></td><td>11:0</td><td><b>DWord Length</b><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><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>0</td><td><b>RefPicList Select</b>
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 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><b>Reference List Entry</b>
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<ul><li>31:24 entry X+3 (e.g. listY_3)</li><li>23:16 entry X+2 (e.g. listY_2)</li><li>15:8 entry X+1 (e.g. listY_1)</li><li>7:0 entry X (e.g. listY_0)</li></ul>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 :<ul><li>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</li><li>Bit 6: Long term bit - set this reference picture to be used as long term reference</li><li>Bit 5: Field picture flag - indicates frame/field</li><li>Bit 4:0: Frame store index or Frame Store ID (Bit 4:1 is used to form the binding table index in this implementation)</li></ul></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 this 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

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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h PARALLEL_VIDEO_PIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Pipeline</b><table><tr><td>Default Value:</td><td>2h MFX_AVC_SLICE_STATE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>26:24</td><td><b>Command Opcode</b><table><tr><td>Default Value:</td><td>1h AVC</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>23:21</td><td><b>SubOpcodeA</b><table><tr><td>Default Value:</td><td>0h MFX_AVC_SLICE_STATE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>20:16</td><td><b>Command SubOpcodeB</b><table><tr><td>Default Value:</td><td>3h MFX_AVC_SLICE_STATE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:12</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>8h DWORD_COUNT_n</td></tr><tr><td>Format:</td><td>=n</td></tr></table>Excludes DWords 0,1</td></tr>
<tr><td>1</td><td>31:4</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>3:0</td><td><b>Slice Type</b>
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></table></td></tr>
</table>

<table>
<tr><th colspan="3">MFX_AVC_SLICE_STATE</th></tr>
<tr><td></td><td></td><td><table><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></td></tr>
<tr><td>2</td><td>31:30</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>29:24</td><td><b>Number of Reference Pictures in Inter-prediction List 1</b><table><tr><td>Format:</td><td>U6</td></tr></table>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></td></tr>
<tr><td></td><td>23:22</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>21:16</td><td><b>Number of Reference Pictures in Inter-prediction List 0</b><table><tr><td>Format:</td><td>U6</td></tr></table>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></td></tr>
<tr><td></td><td>15:11</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>10:8</td><td><b>Log 2 Weight Denom Chroma</b><table><tr><td>Format:</td><td>U3</td></tr></table><table><tr><th>Value</th><th>Name</th></tr><tr><td>0-7</td><td></td></tr></table></td></tr>
<tr><td></td><td>7:3</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>2:0</td><td><b>Log 2 Weight Denom Luma</b><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><b>Weighted Prediction Indicator</b></td></tr>
</table>

MFX_AVC_SLICE_STATE

<table>
<tr><td></td><td></td><td>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><b>Direct Prediction Type</b>
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><b>Disable Deblocking Filter Indicator</b>
<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; filterInternalEdgesFlag is set equal to 1</td></tr>
<tr><td>11b</td><td>Reserved</td><td>Not defined in AVC</td></tr>
</table></td></tr>
<tr><td></td><td>26</td><td><b>Reserved</b></td></tr>
</table>

MFX_AVC_SLICE_STATE

		Format: | MBZ
	25:24	**Cabac Init Idc[1:0]** Specifies the index for determining the initialization table used in the context variable initialization process. **Value** | **Name** 0-2 | **Programming Notes** Cabac initialization is also dependent on the field/frame picture type, Slice type, and the current SliceQP value.
	23:22	**Reserved** Format: | MBZ
	21:16	**Slice Quantization Parameter** Quantization Parameter for current slice. Derived from PPS and slice_delta_qp syntax element in Slice Header. It is needed for CABAC context initialization and deblocking filter control. And it is also used as the starting QP value in the very first MB of a slice. It is in the range of unsigned integer 0 to 51, for 8-bit pixel bit-depth.
	15:12	**Reserved** Format: | MBZ
	11:8	**Slice Beta Offset Div2** Format: | S3 2's Complement Range: [-6, 6] Inclusive Specifies the offset used in accessing the deblocking filter strength tables.
	7:4	**Reserved** Format: | MBZ
	3:0	**Slice Alpha C0 Offset Div2** Format: | S3 2's Complement Range: [-6, 6] Inclusive Specifies the offset used in accessing the deblocking filter strength tables.
4	31:24	**Slice Vertical Position** This field specifies the position in y-direction of the first macroblock in the Slice in unit of macroblocks. The fields (Slice_MB_Start_Hor_Pos, Slice_MB_Start_Vert_Pos) are valid in VLD (decoding) mode only. They are ignored by hardware in decoding IT mode and encoding mode (whereas the position is provided by the per-macroblock object command). Derived **Programming Notes** Error Handling: Driver needs to check if FirstMbY starts at 0 on the first slice of frame. If not,

MFX_AVC_SLICE_STATE

<table>
<tr><td></td><td></td><td><table><tr><td>driver needs to add a phantom slice with FirstMbX and FirstMbY set to 0.</td></tr></table></td></tr>
<tr><td></td><td>23:16</td><td><b>Slice Horizontal Position</b>
This field specifies the position in x-direction of the first macroblock in the Slice in unit of macroblocks.
Derived<table><tr><th>Programming Notes</th></tr><tr><td>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.</td></tr></table></td></tr>
<tr><td></td><td>15</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>14:0</td><td><b>Slice Start Mb Num</b><table><tr><td>Exists If:</td><td>//Decoder Only</td></tr></table>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.<table><tr><th>Programming Notes</th></tr><tr><td>In creating the Phantom Slice for error concealment, this field should set to the total number of MB in the current picture + 1.</td></tr></table></td></tr>
<tr><td>5</td><td>31:24</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>23:16</td><td><b>Next Slice Vertical Position</b>
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).</td></tr>
<tr><td></td><td>15:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>Next Slice Horizontal Position</b>
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.</td></tr>
<tr><td>6
Encoder Only</td><td>31</td><td><b>Rate Control Counter Enable</b>
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.<table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>Disable</td></tr><tr><td>1</td><td>Enable</td></tr></table></td></tr>
<tr><td></td><td>30</td><td><b>ResetRateControlCounter</b>
To reset the bit allocation accumulation counter to 0 to restart the rate control.</td></tr>
</table>

MFX_AVC_SLICE_STATE

Value	Name
0	Not Reset
1	Reset

29:28 **RC Triggle Mode**

Value	Name	Description
00b	Always Rate Control	Whereas RC becomes active if sum_act > sum_target or sum_act < sum_target
01b	Gentle Rate Control	whereas RC becomes active if sum_act > upper_midpt or sum_act < lower_midpt
10b	Loose Rate Control	whereas RC becomes active if sum_act > sum_max or sum_act < sum_min
11b	Reserved	

27:24 **RC Stable Tolerance**

Format:	U4

This field specifies the tolerance required to deactivate RC once it has been triggered.

Value	Name
0-15	

23 **RC Panic Enable**

If this field is set to 1, RC enters panic mode when sum_act > sum_max. RC Panic Type field controls what type of panic behavior is invoked.

Value	Name
0	Disable
1	Enable

22 **RC Panic Type**

This field selects between two RC Panic methods

Value	Name
0	QP Panic
1	CBP Panic

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

21 **MB Type Direct Conversion Disable**

Exists If:	//B-Slice

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

Value	Name

MFX_AVC_SLICE_STATE

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.

16 **SliceData Insertion Present in Bitstream**

Value	Name	Description
0		No Slice Data insertion into the output bitstream buffer
1		Slice Data insertion into the output bitstream buffer is present.

15 **Tail Insertion Present in bitstream**

Value	Name	Description
0		No tail insertion into the output bitstream buffer, after the current slice encoded bits
1		Tail insertion into the output bitstream buffer is present, and is after the current slice encoded bits.

14 **Reserved**

Format:	MBZ

<table>
<tr><th colspan="3">MFX_AVC_SLICE_STATE</th></tr>
<tr><td></td><td>13</td><td><b>EmulationByteSliceInsertEnable</b>
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><b>CabacZeroWordInsertionEnable</b>
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><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table>
For SliceID extension.</td></tr>
<tr><td></td><td>7:4</td><td><b>Slice ID [3:0]</b>
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><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table>
For StreamID extension.</td></tr>
<tr><td></td><td>1:0</td><td><b>Stream ID [1:0]</b>
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><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>28:0</td><td><b>Indirect PAK-BSE Data Start Address (Write)</b>
<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.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0 - 512MB</td><td></td></tr>
</table></td></tr>
<tr><td>8</td><td>31:24</td><td><b>Magnitude of QP Max Negative Modifier</b>
<table>
<tr><td>Format:</td><td>U8</td></tr>
</table></td></tr>
</table>

<table>
<tr><th colspan="3">MFX_AVC_SLICE_STATE</th></tr>
<tr><td rowspan="6">Encoder Only</td><td></td><td>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><b>Magnitude of QP Max Positive Modifier</b><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><b>Shrink Param - Shrink Resistance</b><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><b>Shrink Param - Shrink Init</b><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><b>Grow Param - Grow Resistance</b><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><b>Grow Param - Grow Init</b><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 rowspan="2">9
Encoder Only</td><td>31</td><td><b>RoundInterEnable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>When this bit is not set, RoundInter defaults to 2.</td></tr>
<tr><td>30:28</td><td><b>RoundInter</b><table><tr><td>Format:</td><td>U3</td></tr></table>Rounding precision for Inter quantized coefficients<table><tr><th>Value</th><th>Name</th></tr><tr><td>000b</td><td>+1/16 <b>[Default]</b></td></tr><tr><td>001b</td><td>+2/16</td></tr><tr><td>010b</td><td>+3/16</td></tr></table></td></tr>
</table>

MFX_AVC_SLICE_STATE

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

Value	Name
000b	+1/16 **[Default]**
001b	+2/16
010b	+3/16
011b	+4/16
100b	+5/16
101b	+6/16
110b	+7/16
111b	+8/16

23:20 **Correct 6**

Format:	U4

This field specifies the points used in the lowermost RC region when sum_act <= sum_min.

Value	Name
0 - 15	

19:16 **Correct 5**

Format:	U4

This field specifies the points used in the fifth RC region when sum_act > sum_min but <= lower_midpt.

Value	Name
0 - 15	

15:12 **Correct 4**

Format:	U4

This field specifies the points used in the fourth RC region when sum_act > lower_midpt but <= sum_target.

Value	Name
0 - 15	

11:8 **Correct 3**

MFX_AVC_SLICE_STATE

<table>
<tr><td></td><td></td><td>
<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>
</td></tr>
<tr><td></td><td>7:4</td><td>
<b>Correct 2</b>
<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>
</td></tr>
<tr><td></td><td>3:0</td><td>
<b>Correct 1</b>
<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>
</td></tr>
<tr><td>10
Encoder Only</td><td>31:28</td><td><b>ClampValues - CV7</b></td></tr>
<tr><td></td><td>27:24</td><td><b>CV6</b></td></tr>
<tr><td></td><td>23:20</td><td><b>CV5</b></td></tr>
<tr><td></td><td>19:16</td><td><b>CV4</b></td></tr>
<tr><td></td><td>15:12</td><td><b>CV3</b></td></tr>
<tr><td></td><td>11:8</td><td><b>CV2</b></td></tr>
<tr><td></td><td>7:4</td><td><b>CV1</b></td></tr>
<tr><td></td><td>3:0</td><td>
<b>CV0 - Clamp Value 0</b>
<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).
<b>For 4x4 frame block, each coefficient is mapped to one of the eight CV values as following:</b>
<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>
<b>For 8x8 frame block, each coefficient is mapped to one of the eight CV values as following:</b>
<table>
<tr><td>none</td><td>none</td><td>CV7</td><td>CV6</td><td>CV5</td><td>CV4</td><td>CV3</td><td>CV3</td></tr>
</table>
</td></tr>
</table>

MFX_AVC_SLICE_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 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

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
1	31:1	**Reserved** Format: MBZ
	0	**Weight and Offset Select** It must be set in consistent with the WeightedPredFlag and WeightedBiPredIdc in the Img_State command. This parameter is not present in the DXVA.

<table>
<tr><th colspan="3">MFX_AVC_WEIGHTOFFSET_STATE</th></tr>
<tr><td></td><td></td><td>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><b>WeightOffset</b>
<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>
<tr><td>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].</td></tr>
</table></td></tr>
</table>

MFX_BSP_BUF_BASE_ADDR_STATE

Source:	VideoCS
Length Bias:	2

This frame-level state command is used to specify all the buffer base addresses needed for the operation of the AVC Bit Stream Processing Units (for decoder, it is BSD Unit; for encoder, it is BSE Unit)For both encoder and decoder, currently it is assumed that all codec standards can share the same BSP_BUF_BASE_STATE. The simplicity of this command is the result of moving all the direct MV related processing into the ENC Subsystem. Since all implicit weight calculations and directMV calculations are done in ENC and all picture buffer management are done in the Host, there is no need to provide POC (POC List - FieldOrderCntList, CurrPic POC - CurrFieldOrderCnt) information to PAK. For decoder, all the direct mode information are sent in a separate slice-level command (AVC_DIRECTMODE_STATE command). In addition, in Encoder, the row stores for CABAC encoding and MB Parameters Construction (MPC) are combined into one single row store. The row stores specified in this command do not combine with those specified in the MFC_PIPE_BUF_ADDR_STATE command for hardware simplification reason.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Pipeline Format: OpCode
	26:24	**Media Command Opcode** Default Value: 0h Common Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h MFX_BSP_BUF_BASE_ADDR_STATE Format: OpCode
	20:16	**SubOpcode B** Default Value: 4h MFX_BSP_BUF_BASE_ADDR_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 2h Excludes DWord (0,1) 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

MFX_AVC_WEIGHTOFFSET_STATE

<table>
<tr><td></td><td></td><td><table><tr><td>current macroblock to address this Row Store.</td></tr><tr><td>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.</td></tr></table></td></tr>
<tr><td></td><td>5:4</td><td><b>BSP Row Store Scratch Buffer - Arbitration Priority Control</b><table><tr><td>Format:</td><td>U2 Enumerated Type</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></td><td>30:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>2</td><td>31:6</td><td><b>MPR Row Store Scratch Buffer Base Address - Read/Write (Decoder Only)</b>
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.<table><tr><th>Programming Notes</th></tr><tr><td>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</td></tr></table></td></tr>
<tr><td></td><td>5:4</td><td><b>MPR Row Store Scratch Buffer - Arbitration Priority Control</b><table><tr><td>Format:</td><td>U2 Enumerated type</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><th>Description</th></tr><tr><td>0h</td><td><b>[Default]</b></td><td></td></tr><tr><td>00b</td><td>Highest priority</td><td>Desc</td></tr><tr><td>01b</td><td>Second highest priority</td><td>Desc</td></tr><tr><td>10b</td><td>Third highest priority</td><td></td></tr><tr><td>11b</td><td>Lowest priority</td><td></td></tr></table></td></tr>
<tr><td></td><td>30:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>3</td><td>31:6</td><td><b>Bitplane Read Buffer Base Address</b></td></tr>
</table>

MFX_AVC_WEIGHTOFFSET_STATE

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 Long Format: it is a read-only buffer. For VC1 DXVA2 Short Format: it is a write and a read bufferThis field is only valid for VC1 decoder mode.

5:4 **Bitplane Read Buffer - Arbitration Priority Control**

Format:	U2 Enumerated type

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

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

30:0 **Reserved**

Format:	MBZ

MFX_DBK_OBJECT

Source:	VideoCS
Length Bias:	2

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h PARALLEL_VIDEO_PIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Pipeline</b><table><tr><td>Default Value:</td><td>2h MFX_DBK_OBJECT</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>26:24</td><td><b>Media Command Opcode</b><table><tr><td>Default Value:</td><td>0h Common</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>23:21</td><td><b>SubOpcode A</b><table><tr><td>Default Value:</td><td>0h MEDIA_</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>20:16</td><td><b>SubOpcode B</b><table><tr><td>Default Value:</td><td>9h MEDIA_</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:12</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>3h Excludes DWord (0,1)</td></tr><tr><td>Format:</td><td>=n</td></tr></table><b>Note:</b> Regardless of the mode, inline data must be present in this command</td></tr>
<tr><td>1</td><td>31:6</td><td><b>Pre Deblocking Source Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:6]</td></tr></table>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).</td></tr>
<tr><td></td><td>5:4</td><td><b>Pre Deblocking - Arbitration Priority Control</b>
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><b>Reserved</b></td></tr>
</table>

<table>
<tr><th colspan="3">MFX_DBK_OBJECT</th></tr>
<tr><td></td><td>2</td><td><b>Pre Deblocking - Graphics Data Type (GFDT)</b>
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><b>Pre Deblocking - Cacheability Control</b>
This field controls cacheability.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>00b</td><td>use cacheability control bits from GTT entry</td></tr>
<tr><td>01b</td><td>data is not cached</td></tr>
<tr><td>11b</td><td>data is cached</td></tr>
</table></td></tr>
<tr><td>2</td><td>31:6</td><td><b>Deblocking Control Address</b>
<table>
<tr><td>Format:</td><td>GraphicsAddress[31:6]</td></tr>
</table>
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.</td></tr>
<tr><td></td><td>5:4</td><td><b>Deblocking control - Arbitration Priority Control</b>
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><b>Reserved</b></td></tr>
<tr><td></td><td>2</td><td><b>Deblocking control - Graphics Data Type (GFDT)</b>
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><b>Deblocking control - Cacheability Control</b>
This field controls cacheability.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>00b</td><td>use cacheability control bits from GTT entry</td></tr>
<tr><td>01b</td><td>data is not cached</td></tr>
<tr><td>11b</td><td>data is cached</td></tr>
</table></td></tr>
<tr><td>3</td><td>31:6</td><td><b>Deblocking Destination Address</b>
<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></td><td>5:4</td><td><b>Deblocking - Arbitration Priority Control</b>
This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.</td></tr>
</table>

MFX_DBK_OBJECT

<table>
<tr><td></td><td></td><td><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><b>Reserved</b></td></tr>
<tr><td></td><td>2</td><td><b>Deblocking - Graphics Data Type (GFDT)</b>
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><b>Deblocking - Cacheability Control</b>
This field controls cacheability.<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>use cacheability control bits from GTT entry</td></tr><tr><td>01b</td><td>data is not cached</td></tr><tr><td>11b</td><td>data is cached</td></tr></table></td></tr>
<tr><td>4</td><td>31:6</td><td><b>Deblock Row Store Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:6]</td></tr></table>This field provides the base address of the scratch buffer (read and write) used by the deblocking filter unit to store MB information of the previous row for filtering of each macroblock in the current row. The Deblocking Filter Row Store buffer must be 64-byte cacheline aligned. Hardware uses the horizontal address of the current macroblock to address the Deblocking Filter Row Store.</td></tr>
<tr><td></td><td>5:4</td><td><b>Deblock Row Store - Arbitration Priority Control</b>
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><b>Reserved</b></td></tr>
<tr><td></td><td>2</td><td><b>Deblock Row Store- Graphics Data Type (GFDT)</b>
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>
</table>

<table>
<tr><th colspan="3">MFX_DBK_OBJECT</th></tr>
<tr><td></td><td>1:0</td><td><b>Deblock Row Store - Cacheability Control</b>
This field controls cacheability.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>00b</td><td>use cacheability control bits from GTT entry</td></tr>
<tr><td>01b</td><td>data is not cached</td></tr>
<tr><td>11b</td><td>data is cached</td></tr>
</table>
</td></tr>
</table>

MFX_FQM_STATE

Source:	VideoCS
Length Bias:	2

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_MULTI_DW Format: OpCode
	26:24	**Media Command Opcode** Default Value: 0h MFX_COMMON_STATE Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 8h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 20h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:2	**Reserved** Format: MBZ
	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

MFX_FQM_STATE

<table>
<tr><td></td><td>1:0</td><td><b>AVC</b>
<table>
<tr><td>Exists If:</td><td>//AVC- Decoder Only</td></tr>
</table>
<b>For AVC QM Type</b>: This field specifies which Quantizer Matrix is loaded.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0</td><td>AVC_4x4_Intra_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs)</td></tr>
<tr><td>1</td><td>AVC_4x4_Inter_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs)</td></tr>
<tr><td>2</td><td>AVC_8x8_Intra_MATRIX</td></tr>
<tr><td>3</td><td>AVC_8x8_Inter_MATRIX</td></tr>
</table>
</td></tr>
<tr><td>2..33</td><td>31:0</td><td><b>Forward Quantizer Matrix</b>
<table>
<tr><td>Format:</td><td>U32</td></tr>
</table>
The format of a Quantizer Matrix is an 8x8 matrix in raster order. Each element is an unsigned byte.
</td></tr>
</table>

MFX_IND_OBJ_BASE_ADDR_STATE

Source:	VideoCS
Length Bias:	2

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

The MFX_IND_OBJ_BASE_ADDR command sets the memory base address pointers for the corresponding Indirect Object Data Start Addresses (Offsets) specified in each OBJECT commands. The characteristic of these indirect object data is their variable size (per MB or per Slice). Hence, each OBJECT command must specify the indirect object data offset from the base address to start fetching or writing object data.

While the use of base address is unconditional, the indirection can be effectively disabled by setting the base address to zero. For decoder, there are only 1 read-only per-slice indirect object in the BSD_OBJECT Command, and 2 read-only per-MB indirect objects in the IT_OBJECT CommandFor decoder: the Video Command Streamer (VCS) will perform the memory access bound check automatically using the corresponding MFC Indirect Object Access Upper Bound specification. If any access is at or beyond the upper bound, zero value is returned. The request to memory is still being sent, but the corresponding codec's BSD unit will detect this condition and perform the zeroing return. If the Upper Bound is turned off, the beyond bound request will return whatever on the bus (invalid data). For encoder, there are 1 read-only per-MB indirect object in the PAK_OBJECT Command, and 1 write-only per-slice indirect object in the PAK Slice_State CommandFor encoder: whenever an out of bound address accessing request is generated, VMX will detect such requests and snap the address to the corresponding [indirect object base address + indirect data start address]. VMX will return all 0s as the data to the requester. 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).

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h PARALLEL_VIDEO_PIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Pipeline</b><table><tr><td>Default Value:</td><td>2h MFX_IND_OBJ_BASE_ADDR_STATE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>26:24</td><td><b>Common Opcode</b><table><tr><td>Default Value:</td><td>0h MFX_IND_OBJ_BASE_ADDR_STATE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>23:21</td><td><b>Sub OpcodeA</b><table><tr><td>Default Value:</td><td>0h MFX_IND_OBJ_BASE_ADDR_STATE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>20:16</td><td><b>SubOpcodeB</b></td></tr>
</table>

<table>
<tr><th colspan="3">MFX_IND_OBJ_BASE_ADDR_STATE</th></tr>
<tr><td></td><td></td><td><table><tr><td>Default Value:</td><td>3h MFX_IND_OBJ_BASE_ADDR_STATE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:12</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>0009h Excludes DWord (0,1)</td></tr><tr><td>Format:</td><td>=n Total Length - 2</td></tr></table></td></tr>
<tr><td>1</td><td>31:12</td><td><b>MFX Indirect Bitstream Object - Base Address (Decoder and Stitch Modes)</b><table><tr><td>Format:</td><td>GraphicsAddress[31:12]</td></tr></table>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.</td></tr>
<tr><td></td><td>11:6</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>5:4</td><td><b>MFX Indirect BSD Object - Arbitration Priority Control</b><table><tr><td>Format:</td><td>U2 Enumerated Type</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></td><td>3</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>2</td><td><b>MFX Indirect Bitstream Object - Graphics Data Type (GFDT)</b><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.</td></tr>
<tr><td></td><td>1:0</td><td><b>MFX Indirect Bitstream Object - Cacheability Control</b><table><tr><td>Format:</td><td>U2 Enumerated Type</td></tr></table>This field controls cacheability.<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 cached</td><td>data is not cached</td></tr><tr><td>11b</td><td>Cached</td><td>data is cached</td></tr></table></td></tr>
<tr><td>2</td><td>31:12</td><td><b>MFX Indirect Bitstream Object - Access Upper Bound (Decoder and Stitch Modes)</b><table><tr><td>Format:</td><td>GraphicsAddress[31:12]</td></tr></table></td></tr>
</table>

<table>
<tr><th colspan="3">MFX_IND_OBJ_BASE_ADDR_STATE</th></tr>
<tr><td></td><td></td><td>This field specifies the 4K-byte aligned (exclusive) maximum Graphics Memory address access by the indirect data object in the MFD_XXX_BSD_OBJECT command for the compressed Slice Data. Indirect data accessed at this address and beyond will return as 0 by the hardware. Setting this field to 0 will cause this range check to be ignored. If non-zero, this address must be greater than the MFX Indirect Bitstream ObjectBase Address state. Hardware ignores this field if indirect data is not present, i.e. the Indirect Data Length field of the MFD_XXX_BSD_OBJECT command is set to 0.This field is only valid in MPEG2, AVC and VC1 decoder VLD mode.</td></tr>
<tr><td></td><td>11:0</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>3</td><td>31:12</td><td><b>MFX Indirect MV Object - Base Address</b>
<table><tr><td>Format:</td><td>GraphicsAddress[31:12]</td></tr></table>
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</td></tr>
<tr><td></td><td>11:6</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>5:4</td><td><b>MFX Indirect MV Object - Arbitration Priority Control</b>
<table><tr><td>Format:</td><td>U2 Enumerated Type</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></td><td>3</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>2</td><td><b>MFX Indirect MV Object - Graphics Data Type (GFDT)</b>
<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.</td></tr>
<tr><td></td><td>1:0</td><td><b>MFX Indirect MV Object - Cacheability Control</b>
<table><tr><td>Format:</td><td>U2</td></tr></table>
This field controls cacheability.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>00b</td><td>From GTT entry</td><td>use cacheability control bits from GTT entry</td></tr>
<tr><td>01b</td><td>Not cached</td><td>data is not cached</td></tr>
<tr><td>11b</td><td>Cached</td><td>data is cached</td></tr>
</table></td></tr>
</table>

<table>
<tr><th colspan="3">MFX_IND_OBJ_BASE_ADDR_STATE</th></tr>
<tr><td>4</td><td>31:12</td><td><b>MFX Indirect MV Object Access Upper Bound</b>
<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 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.</td></tr>
<tr><td></td><td>11:0</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>5</td><td>31:12</td><td><b>MFD Indirect IT-COEFF Object - Base Address (Decoder Only)</b>
<table><tr><td>Format:</td><td>GraphicsAddress[31:12]</td></tr></table>
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.</td></tr>
<tr><td></td><td>11:6</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>5:4</td><td><b>MFD Indirect IT-COEFF Object - Arbitration Priority Control</b>
<table><tr><td>Format:</td><td>U2 Enumerated Type</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></td><td>3</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>2</td><td><b>MFD Indirect IT-COEFF Object - Graphics Data Type (GFDT)</b>
<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.</td></tr>
<tr><td></td><td>1:0</td><td><b>MFD Indirect IT-COEFF Object - Cacheability Control</b>
<table><tr><td>Format:</td><td>U2 Enumerated type</td></tr></table>
This field controls cacheability.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>00b</td><td>From GTT entry</td><td>use cacheability control bits from GTT entry</td></tr>
</table></td></tr>
</table>

MFX_IND_OBJ_BASE_ADDR_STATE

<table>
<tr><td></td><td></td><td><table><tr><td>01b</td><td>Not cached</td><td>data is not cached</td></tr><tr><td>11b</td><td>Cached</td><td>data is cached</td></tr></table></td></tr>
<tr><td>6</td><td>31:12</td><td><b>MFD Indirect IT-COEFF Object - Access Upper Bound (Decoder Only)</b><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_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.</td></tr>
<tr><td></td><td>11:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>7</td><td>31:12</td><td><b>MFD Indirect IT-DBLK Object - Base Address (Decoder Only)</b><table><tr><td>Format:</td><td>GraphicsAddress[31:12]</td></tr></table>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.</td></tr>
<tr><td></td><td>11:6</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>5:4</td><td><b>MFD Indirect IT-DBLK Object - Arbitration Priority Control</b><table><tr><td>Format:</td><td>U2 Enumerated Type</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></td><td>3</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>2</td><td><b>MFD Indirect IT-DBLK Object - Graphics Data Type (GFDT)</b><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.</td></tr>
<tr><td></td><td>1:0</td><td><b>MFD Indirect IT-DBLK Object - Cacheability Control</b><table><tr><td>Format:</td><td>U2 Enumerated Type</td></tr></table>This field controls cacheability.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr></table></td></tr>
</table>

MFX_IND_OBJ_BASE_ADDR_STATE

<table>
<tr><td></td><td></td><td><table><tr><td>00b</td><td>From GTT entry</td><td>use cacheability control bits from GTT entry</td></tr><tr><td>01b</td><td>Not cached</td><td>data is not cached</td></tr><tr><td>11b</td><td>Cached</td><td>data is cached</td></tr></table></td></tr>
<tr><td>8</td><td>31:12</td><td><b>MFD Indirect IT-DBLK Object Access Upper Bound (Decoder Only)</b><table><tr><td>Format:</td><td>GraphicsAddress[31:12]</td></tr><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_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.</td></tr>
<tr><td></td><td>11:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>9</td><td>31:12</td><td><b>MFC Indirect PAK-BSE Object - Base Address (Encoder Only)</b><table><tr><td>Format:</td><td>GraphicsAddress[31:12]</td></tr></table>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.</td></tr>
<tr><td></td><td>11:6</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>5:4</td><td><b>MFC Indirect PAK-BSE Object - Arbitration Priority Control</b><table><tr><td>Format:</td><td>U2 Enumerated Type</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></td><td>3</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>2</td><td><b>MFC Indirect PAK-BSE Object - Graphics Data Type (GFDT)</b><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.</td></tr>
<tr><td></td><td>1:0</td><td><b>MFC Indirect PAK-BSE Object - Cacheability Control</b><table><tr><td>Format:</td><td>U2 Enumerated Type</td></tr></table></td></tr>
</table>

<table>
<tr><th colspan="3">MFX_IND_OBJ_BASE_ADDR_STATE</th></tr>
<tr><td></td><td></td><td>This field controls cacheability.
<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 cached</td><td>data is not cached</td></tr>
<tr><td>11b</td><td>Cached</td><td>data is cached</td></tr>
</table>
</td></tr>
<tr><td>10</td><td>31:12</td><td><b>MFC Indirect PAK-BSE Object - Access Upper Bound (Encoder Only)</b>
<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 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.</td></tr>
<tr><td></td><td>11:0</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table>
</td></tr>
</table>

MFX_JPEG_HUFF_TABLE_STATE

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** Format: MBZ
	11:0	**DWord Length** Default Value: 033Dh Excludes DWord (0,1) Format: =n Total Length - 2
1	31:1	**Reserved** Format: MBZ
	0	**HuffTableID (1-bit)** Identifies the huffman table. Value | Name | Description 0 | Y | Huffman table for Y
2..4	31:0	**DC_BITS (12 8-bit array)** The number of DC Huffman codes of length i, where i is 1~12
5..7	31:0	**DC_HUFFVAL (12 8-bit array)** The value associated with each DC Huffman code of length i.
8..11	31:0	**AC_BITS (16 8-bit array)** the list of Li, number of Huffman codes of length i, where i is 1~16

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

MFX_JPEG_PIC_STATE

Source: VideoCS

Length Bias: 2

DWord	Bit	Description		
0	31:29	**Command Type**		
		Default Value:	3h PARALLEL_VIDEO_PIPE	
		Format:	OpCode	
	28:27	**Pipeline**		
		Default Value:	2h 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**		
		Format:	=n Total Length - 2	
		Value	**Name**	**Description**
		0001h	**[Default]**	Excludes DWord (0,1)
1	31:21	**Reserved**		
		Exists If:	//Decoder Only	
		Format:	MBZ	
	20:19	**Reserved**		
		Exists If:	//Decoder Only	
		Format:	MBZ	
	18	**Reserved**		
		Exists If:	//Decoder Only	
		Format:	MBZ	
	17:16	**Reserved**		
		Exists If:	//Decoder Only	
		Format:	MBZ	

MFX_JPEG_PIC_STATE

DWord	Bit	Description
	15:12	**Reserved**

Exists If:	//Decoder Only
Format:	MBZ

DWord	Bit	Description
	11:8	**Reserved**

Exists If:	//Decoder Only
Format:	MBZ

DWord	Bit	Description
	7:6	**Reserved**

Exists If:	//Decoder Only
Format:	MBZ

DWord	Bit	Description
	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)
11b		rotate 180 degree (NOT the same as flipped on the x-axis)

DWord	Bit	Description
	3	**Reserved**

Exists If:	//Decoder Only
Format:	MBZ

DWord	Bit	Description
	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

DWord	Bit	Description
2	31:30	**Reserved**

Exists If:	//Decoder Only
Format:	MBZ

MFX_JPEG_PIC_STATE

	29	**Reserved** Exists If: //Decoder Only Format: MBZ
	28:16	**Frame Height In Blocks Minus 1** Exists If: //Decoder 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 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, $((Y + 7) / 8) - 1$.
	15:13	**Reserved** Exists If: //Decoder Only Format: MBZ
	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, $((X + 7) / 8) - 1$.

MFX_MPEG2_PIC_STATE

Source:	VideoCS
Length Bias:	2

This must be the very first command to issue after the surface state, the pipe select and base address setting commands. For MPEG-2 the encoder is called per slice-group, however the picture state is called per picture. Notice that a slice-group is a group of consecutive slices that no non-trivial slice headers are inserted in between.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_MPEG2_PIC_STATE Format: OpCode
	26:24	**Media Command Opcode** Default Value: 3h MPEG2_COMMON Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 0h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 0h Excludes DWord (0,1)= 00Bh, used for normal decode and encode mode000h, a special case to provide a dummy image state for stitch mode operation. In this case, fields in DW1 which is part of the dummy image state command are ignored by hardware. Format: =n Total Length - 2
1	31:28	**f_code[1][1].** Used for backward motion vector prediction. See ISO/IEC 13818-2 7.6.3.1 for details
	27:24	**f_code[1][0].** Used for backward motion vector prediction. See ISO/IEC 13818-2 7.6.3.1 for details
	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**

<table>
<tr><th colspan="3">MFX_MPEG2_PIC_STATE</th></tr>
<tr><td></td><td></td><td><table><tr><td>Format:</td><td>U2</td></tr></table>See ISO/IEC 13818-2 6.3.10 for details.</td></tr>
<tr><td></td><td>13:12</td><td><b>Picture Structure</b>
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</td></tr>
<tr><td></td><td>11</td><td><b>TFF (Top Field First)</b>
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</td></tr>
<tr><td></td><td>10</td><td><b>Frame Prediction Frame DCT</b>
This field provides constraints on the DCT type and prediction type. It affects the syntax of the bitstream.</td></tr>
<tr><td></td><td>9</td><td><b>Concealment Motion Vector Flag</b>
This field indicates if the concealment motion vectors are coded in intra macroblocks. It affects the syntax of the bitstream.</td></tr>
<tr><td></td><td>8</td><td><b>Quantizer Scale Type</b><table><tr><td>Format:</td><td>MPEG_Q_SCALE_TYPE</td></tr></table>This field specifies the quantizer scaling type.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td></td><td>MPEG_QSCALE_LINEAR</td></tr><tr><td>1h</td><td></td><td>D MPEG_QSCALE_NONLINEAR esc</td></tr></table></td></tr>
<tr><td></td><td>7</td><td><b>Intra VLC Format</b>
This field is used by VLD</td></tr>
<tr><td></td><td>6</td><td><b>Scan Order</b><table><tr><td>Format:</td><td>MPEG_INVERSESCAN_TYPE</td></tr></table>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><b>Reserved</b></td></tr>
<tr><td>2</td><td>31:24</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
</table>

MFX_MPEG2_PIC_STATE

23:15 **Reserved**

Format:	MBZ

14 **LoadSlicePointerFlag - LoadBitStreamPointerPerSlice**

Exists If:	//Encoder

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

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

13 **Reserved**

Format:	MBZ

was Concealment Enable

12 **Reserved**

Format:	MBZ

was Concealment Reference

11 **Reserved**

Format:	MBZ

was Concealment Type

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

was Slice Error Control

<table>
<tr><th colspan="3">MFX_MPEG2_PIC_STATE</th></tr>
<tr><td></td><td>1</td><td><b>MismatchControlDisabled</b>
<table>
<tr><th>Description</th></tr>
<tr><td>These 2 bits flag disables mismatch control of the inverse transformation for some specific cases during reference reconstruction.</td></tr>
</table>
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>00b</td><td></td><td>Mismatch control applies to all MBs</td></tr>
<tr><td>01b</td><td></td><td>Disable mismatch control to all intra MBs whose all AC-coefficients are zero.</td></tr>
<tr><td>10b</td><td></td><td>Disable mismatch control to all MBs whose all AC-coefficients are zero.</td></tr>
<tr><td>11b</td><td></td><td>Disable mismatch control to all MBs.</td></tr>
</table></td></tr>
<tr><td></td><td>0</td><td><b>Disable Mismatch</b>
To disable MPEG2 IDCT fixed point arithmetic correction</td></tr>
<tr><td>3</td><td>31</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>30:29</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>28:24</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>23:16</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>15:8</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ for future supporting width &gt; 4K</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>Reserved</b></td></tr>
<tr><td>4</td><td>31:16</td><td><b>MinFrameWSize</b>
<table><tr><td>Format:</td><td>U16</td></tr></table>
- Minimum Frame Size [15:0] (16-bit) (Encoder Only)Mininum Frame Size is specified to compensate for 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. 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><b>[Default]</b></td><td></td></tr>
</table></td></tr>
<tr><td></td><td>15</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>14:12</td><td><b>RoundInterAC,</b></td></tr>
</table>

MFX_MPEG2_PIC_STATE

<table>
<tr><td></td><td></td><td>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><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>10:8</td><td><b>RoundIntraAC</b><table><tr><td>Format:</td><td>U3</td></tr></table>rounding precision for Intra AC000: +1/16001: +2/16010: +3/16011: +4/16100: +5/16101: +6/16110: +7/16111: +8/16</td></tr>
<tr><td></td><td>7</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>6:4</td><td><b>RoundInterDC</b>
rounding Precision for non-Intra-DC000: +1/16001: +2/16010: +3/16011: +4/16100: +5/16101: +6/16110: +7/16111: +8/16</td></tr>
<tr><td></td><td>3</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>2:1</td><td><b>RoundIntraDC</b>
rounding Precision for Intra-DC00: +1/801: +2/810: +3/811: +4/8</td></tr>
<tr><td></td><td>0</td><td><b>Reserved</b></td></tr>
<tr><td>5</td><td>31:17</td><td><b>Reserved</b>
(for future Mask bits)</td></tr>
<tr><td></td><td>16</td><td><b>FrameSizeControlMask</b>
Frame size conformance maskThis field is used when MacroblockStatEnable is set to 1.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td></td><td>Do not change Slice Quantization Parameter values in MFC_MPEG2_SLICEGROUP_STATE with suggested slice QP value for frame level Rate control</td></tr><tr><td>1h</td><td></td><td>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.</td></tr></table></td></tr>
<tr><td></td><td>15:13</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>12</td><td><b>InterMBForceCBPZeroControlMask</b><table><tr><td>Format:</td><td>U1</td></tr></table>Inter MB Force CBP ZERO mask.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>[0,FFFFFFFFh]</td><td></td><td></td></tr><tr><td>0h</td><td></td><td>No effect</td></tr><tr><td>1h</td><td></td><td>Zero out all A/C coefficients for the inter MB violating Inter Confirmance</td></tr></table></td></tr>
<tr><td></td><td>11:10</td><td><b>MinFrameWSizeUnits</b>
This field is the Minimum Frame Size Units</td></tr>
</table>

MFX_MPEG2_PIC_STATE

<table>
<tr><td></td><td></td><td><table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>00b</td><td>compatibility mode</td><td>Minimum Frame Size is in old mode (words, 2bytes)</td></tr><tr><td>01b</td><td>16 byte</td><td>Minimum Frame Size is in 16bytes</td></tr><tr><td>10b</td><td>4Kb</td><td>Minimum Frame Size is in 4Kbytes</td></tr><tr><td>11b</td><td>16Kb</td><td>Minimum Frame Size is in 16Kbytes</td></tr></table></td></tr>
<tr><td></td><td>9</td><td><b>MBRateControlMask</b>
MB Rate Control conformance maskThis field is ignored when MacroblockStatEnable is disabled or MB level Rate control flag for the current MB is disable in Macroblock Status Buffer.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td></td><td>Do not change QP values of inter macroblock with suggested QP values in Macroblock Status Buffer</td></tr><tr><td>1h</td><td></td><td>Apply RC QP delta for all macroblock</td></tr></table></td></tr>
<tr><td></td><td>8</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>7</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>6:4</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>3</td><td><b>FrameBitRateMinReportMask</b>
This is a mask bit controlling if the condition of frame level bit count is less than FrameBitRateMin.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Disable</td><td>Do not update bit0 of MFC_IMAGE_STATUS control register.</td></tr><tr><td>1h</td><td>Enable</td><td>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.</td></tr></table></td></tr>
<tr><td></td><td>2</td><td><b>FrameBitRateMaxReportMask</b>
This is a mask bit controlling if the condition of frame level bit count exceeds FrameBitRateMax.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Disable</td><td>Do not update bit0 of MFC_IMAGE_STATUS control register.</td></tr><tr><td>1h</td><td>Enable</td><td>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.</td></tr></table></td></tr>
<tr><td></td><td>1</td><td><b>InterMBMaxSizeReportMask</b>
This is a mask bit controlling if the condition of any inter MB in the frame exceeds InterMBMaxSize.<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 Inter MB Conformance Max size</td></tr></table></td></tr>
</table>

<table>
<tr><th colspan="3">MFX_MPEG2_PIC_STATE</th></tr>
<tr><td></td><td></td><td><table><tr><td></td><td></td><td>limit.</td></tr></table></td></tr>
<tr><td></td><td>0</td><td><b>IntraMBMaxSizeReportMask</b>
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>6
[ExistsIf]Encode Only</td><td>31:28</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>27:16</td><td><b>InterMBMaxSize</b><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></td><td>15:12</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11:0</td><td><b>IntraMBMaxSize</b><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>7</td><td>31:1</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>8
[ExistsIf]Encode Only</td><td>31:24</td><td><b>SliceDeltaQPMax[3]</b><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><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>
<tr><td></td><td>23:16</td><td><b>SliceDeltaQPMax[2]</b></td></tr>
</table>

MFX_MPEG2_PIC_STATE

		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).
	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

MFX_MPEG2_PIC_STATE

<table>
<tr><td></td><td></td><td><table><tr><td>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)).</td></tr></table></td></tr>
<tr><td></td><td>15:8</td><td><b>SliceDeltaQPMin[1]</b><table><tr><td>Format:</td><td>S7</td></tr></table><table><tr><td>Range: [-30,30]</td></tr><tr><td>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)).</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>SliceDeltaQPMin[0]</b><table><tr><td>Format:</td><td>S7</td></tr></table><table><tr><td>Range: [-30,30]</td></tr><tr><td>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).</td></tr></table></td></tr>
<tr><td>10
[ExistsIf]Encode Only</td><td>31</td><td><b>FrameBitrateMaxUnit</b>
This field is the Frame Bitrate Maximum Limit Units.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Byte</td><td>FrameBitRateMax is in units of 32 Bytes when FrameBitrateMaxUnitMode is 1 and in units of 128 Bytes if FrameBitrateMaxUnitMode is 0</td></tr><tr><td>1h</td><td>Kilobyte</td><td>FrameBitRateMax is in units of 4KBytes Bytes when FrameBitrateMaxUnitMode is 1 and in units of 16KBytes if FrameBitrateMaxUnitMode is 0</td></tr></table></td></tr>
<tr><td></td><td>30</td><td><b>FrameBitrateMaxUnitMode</b>
BitFiel This field is the Frame Bitrate Maximum Limit Units.dDesc<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Compatibility mode</td><td>FrameBitRateMaxUnit is in old mode (128b/16Kb)</td></tr><tr><td>1h</td><td>New mode</td><td>FrameBitRateMaxUnit is in new mode (32byte/4Kb)</td></tr></table></td></tr>
<tr><td></td><td>29:16</td><td><b>FrameBitRateMax</b>
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</td></tr>
</table>

<table>
<tr><th colspan="3">MFX_MPEG2_PIC_STATE</th></tr>
<tr><td></td><td></td><td>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.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0-512KB</td><td></td><td>The programmable range 0-512KB when FrameBitrateMaxUnit is 0.</td></tr>
<tr><td>0-8190KB</td><td></td><td>The programmable range 0-8190KB when FrameBitrateMaxUnit is 1.</td></tr>
</table></td></tr>
<tr><td></td><td>15</td><td><b>FrameBitrateMinUnit</b>
This field is the Frame Bitrate Minimum Limit Units.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Byte</td><td>FrameBitRateMax is in units of 32 Bytes when FrameBitrateMinUnitMode is 1 and in units of 128 Bytes if FrameBitrateMinUnitMode is 0</td></tr>
<tr><td>1h</td><td>KiloByte</td><td>FrameBitRateMax is in units of 4KBytes Bytes when FrameBitrateMaxUnitMode is 1 and in units of 16KBytes if FrameBitrateMaxUnitMode is 0</td></tr>
</table></td></tr>
<tr><td></td><td>14</td><td><b>FrameBitrateMinUnitMode</b>
This field is the Frame Bitrate Minimum Limit Units.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>compatibility mode</td><td>FrameBitRateMaxUnit is in old mode (128b/16Kb)</td></tr>
<tr><td>1h</td><td>New Mode</td><td>FrameBitRateMaxUnit is in new mode (32byte/4Kb)</td></tr>
</table></td></tr>
<tr><td></td><td>13:0</td><td><b>FrameBitRateMin</b>
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</td></tr>
<tr><td>11

[ExistsIf]Encode Only</td><td>31</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>30:16</td><td><b>FrameBitRateMaxDelta</b>
<table>
<tr><td>Default Value:</td><td>0h</td></tr>
<tr><td>Access:</td><td>None</td></tr>
<tr><td>Format:</td><td>U15</td></tr>
</table>
<table>
<tr><td>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.</td></tr>
<tr><td>This field is used to select the slice delta QP when FrameBitRateMax Is exceeded. It</td></tr>
</table></td></tr>
</table>

MFX_MPEG2_PIC_STATE

<table>
<tr><td></td><td></td><td>shares the same FrameBitrateMaxUnit. When FrameBitrateMaxUnitMode is 0(compatibility mode) bits 16:27 should be used, bits 28, 29 and 30 should be 0.</td></tr>
<tr><td></td><td>15</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>14:0</td><td><b>FrameBitRateMinDelta</b>
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.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0-1024KB</td><td></td><td>The programmable range 0-1024KB When FrameBitrateMinUnit is in 32Bytes.</td></tr><tr><td>0-16380KB</td><td></td><td>Programmable range is 0-16380KB when FrameBitrateMinUnit is in 4Kbytes.</td></tr></table></td></tr>
<tr><td>12</td><td>31:21</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>20</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>19</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>18:16</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>15:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
</table>

MFX_PAK_INSERT_OBJECT

Source:	VideoCS
Length Bias:	2

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

MFX_PAK_INSERT_OBJECT

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><b>Command Type</b><table><tr><td>Default Value:</td><td>3h PARALLEL_VIDEO_PIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Pipeline</b><table><tr><td>Default Value:</td><td>2h MFX_PAK_INSERT_OBJECT</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>26:24</td><td><b>Media Command Opcode</b><table><tr><td>Default Value:</td><td>0h MFX_COMMON</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>23:21</td><td><b>SubOpcode A</b><table><tr><td>Default Value:</td><td>2h</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>20:16</td><td><b>SubOpcode B</b><table><tr><td>Default Value:</td><td>8h</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:12</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>0h Excludes DWord (0,1) = Variable Length in DW</td></tr><tr><td>Format:</td><td>=n Total Length - 2</td></tr></table></td></tr>
<tr><td>1</td><td>31:18</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>17:16</td><td><b>DataByteOffset - SrcDataStartingByteOffset[1:0]</b>
Source Data Starting Byte Position within the very first inline DW.</td></tr>
<tr><td></td><td>15</td><td><b>HeaderLengthExcludeFrmSize</b>
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></table></td></tr>
</table>

MFX_PAK_INSERT_OBJECT

1	NO_ACCUMULATION	Bits during current call are not accumulated
0	ACCUMULATE	All bits accumulated

14 **Slice Header Indicator**

This bit indicates if the insert object is a slice header.

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
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.

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.

2 **LastHeaderFlag - LastSrcHeaderDataInsertCommandFlag**

To process a series of consecutive insertion commands, this flag (=1) indicates the current command is the last 'header' insertion in the series. In CABAC, hardware must perform the "1" insert for byte align for Slice Header before Slice Data comes in in the next PAK-OBJECT command. In CAVLC, hardware ignores this bit

1 **EndOfSliceFlag - LastDstDataInsertCommandFlag**

No more insertion command and no more PAK-OBJECT command follows. Flush data out to

MFX_PAK_INSERT_OBJECT

<table>
<tr><td></td><td></td><td>memory</td></tr>
<tr><td></td><td>0</td><td><b>BitstreamStartReset - ResetBitStreamStartingPos</b>
OPEN: This bit is redundant, the control is already in the Slice State command
<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>
</td></tr>
<tr><td>2..n</td><td>31:0</td><td><b>Insert Data PayLoad</b>
Actual Data to be inserted to the output bitstream buffer.</td></tr>
</table>

MFX_PIPE_BUF_ADDR_STATE

Source:	VideoCS
Length Bias:	2

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_PIPE_BUF_ADDR_STATE Format: OpCode
	26:24	**Common Opcode** Default Value: 0h MFX_PIPE_BUF_ADDR_STATE Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h MFX_PIPE_BUF_ADDR_STATE Format: OpCode
	20:16	**SubOpcode B** Default Value: 2h MFX_PIPE_BUF_ADDR_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Total Length Fixed Length Value | Name | Description 16h | DWORD_COUNT_n **[Default]** | Excludes DWord (0,1)
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

MFX_PIPE_BUF_ADDR_STATE

<table>
<tr><td></td><td></td><td>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).</td></tr>
<tr><td></td><td>5:4</td><td><b>Pre Deblocking - Arbitration Priority Control</b>
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>3:0</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td>2</td><td>31:6</td><td><b>Post Deblocking - Destination Address</b>
<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)This field is ignored if PostDeblockOutEnable is set to 0 (disable).</td></tr>
<tr><td></td><td>5:4</td><td><b>Post Deblocking - Arbitration Priority Control</b>
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><b>Reserved</b></td></tr>
<tr><td></td><td>2</td><td><b>Post Deblocking - Graphics Data Type (GFDT)</b>
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><b>Post Deblocking - Cacheability Control</b>
<table>
<tr><td>Format:</td><td>U2 Enumerated type</td></tr>
</table>
This field controls cacheability.
<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 cached</td><td>Data is not cached</td></tr>
<tr><td>11b</td><td>Cached</td><td>Data is cached</td></tr>
</table></td></tr>
<tr><td>3</td><td>31:6</td><td><b>Original Uncompressed Picture - Source Address (CurSrcAddr)</b>
<table>
<tr><td>Exists If:</td><td>//Encoding</td></tr>
<tr><td>Format:</td><td>GraphicsAddress[31:6]</td></tr>
</table></td></tr>
</table>

<table>
<tr><th colspan="3">MFX_PIPE_BUF_ADDR_STATE</th></tr>
<tr><td></td><td></td><td>Specifies the 64 byte aligned frame buffer address for fetching YUV pixel data from the original uncompressed input picture for encoding.</td></tr>
<tr><td></td><td>5:4</td><td><b>Original Uncompressed Picture - Arbitration Priority Control</b>
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><b>Reserved</b></td></tr>
<tr><td></td><td>2</td><td><b>Original Uncompressed Picture - Graphics Data Type (GFDT)</b>
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><b>Original Uncompressed Picture - Cacheability Control</b>
This field controls cacheability.
<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 cached</td><td>data is not cached</td></tr>
<tr><td>11b</td><td>Cached</td><td>Data is cached</td></tr>
</table></td></tr>
<tr><td>4</td><td>31:6</td><td><b>StreamOut Data Destination - Base Address (StreamOutAddr)</b>
<table>
<tr><td>Format:</td><td>GraphicsAddress[31:6]</td></tr>
</table>
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.</td></tr>
<tr><td></td><td>5:4</td><td><b>StreamOut Data Destination - Arbitration Priority Control</b>
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><b>Reserved</b></td></tr>
<tr><td></td><td>2</td><td><b>StreamOut Data Destination - Graphics Data Type (GFDT)</b>
This field contains the GFDT bit for this surface when writes occur. GFDT can also be set by the</td></tr>
</table>

MFX_PIPE_BUF_ADDR_STATE

<table>
<tr><td></td><td></td><td>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><b>StreamOut Data Destination - Cacheability Control</b>
This field controls cacheability.<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 cached</td><td>data is not cached</td></tr>
<tr><td>11b</td><td>Cached</td><td>Data is cached</td></tr>
</table></td></tr>
<tr><td>5</td><td>31:6</td><td><b>Intra Row Store Scratch Buffer - Base Address (IntraOSRowStoreAddr)</b><table>
<tr><td>Format:</td><td>GraphicsAddress[31:6]</td></tr>
</table>This field provides the base address of the scratch buffer (read/write) used by the AVC IntraPrediction unit to store MB information of the previous row for processing of each macroblock in the current row. The Intra Row Store buffer must be 64-byte cacheline aligned. Hardware uses the horizontal address of the current macroblock to address the Intra Row Store. This field is ignored in MPEG2 and VC1 mode. Max 256 cachelines for 4K pixels (1 cacheline for either MBAFF or non-MBAFF)</td></tr>
<tr><td></td><td>5:4</td><td><b>Intra/Overlap Smoothing Row Store Scratch Buffer - Arbitration Priority Control</b>
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><b>Reserved</b></td></tr>
<tr><td></td><td>2</td><td><b>Intra/Overlap Smoothing Row Store Scratch Buffer - Graphics Data Type (GFDT)</b>
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><b>Intra/Overlap Smoothing Row Store Scratch Buffer - Cacheability Control</b>
This field controls cacheability.<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 cached</td><td>data is not cached</td></tr>
<tr><td>11b</td><td>Cached</td><td>Data is cached</td></tr>
</table></td></tr>
<tr><td>6</td><td>31:6</td><td><b>Deblocking Filter Row Store Scratch Buffer - Base Address (DeblockRowStoreAddr</b><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
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</td></tr>
</table>

<table>
<tr><th colspan="3">MFX_PIPE_BUF_ADDR_STATE</th></tr>
<tr><td></td><td></td><td>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).</td></tr>
<tr><td></td><td>5:4</td><td><b>Deblocking Filter Row Store Scratch Buffer - Arbitration Priority Control</b>
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>0h</td><td>Highest priority</td></tr>
<tr><td>1h</td><td>Second highest priority</td></tr>
<tr><td>2h</td><td>Third highest priority</td></tr>
<tr><td>3h</td><td>Lowest priority</td></tr>
</table></td></tr>
<tr><td></td><td>3</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>2</td><td><b>Deblocking Filter Row Store Scratch Buffer - Graphics Data Type (GFDT)</b>
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><b>Deblocking Filter Row Store Scratch Buffer - Cacheability Control</b>
This field controls cacheability.
<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 cached</td><td>data is not cached</td></tr>
<tr><td>11b</td><td>Cached</td><td>Data is cached</td></tr>
</table></td></tr>
<tr><td>7..22</td><td>31:6</td><td><b>Reference Picture (RefAddr[0-15]) - Addresses</b>
<table>
<tr><td>Format:</td><td>GraphicsAddress[31:6]</td></tr>
</table>
Specifies the 64 byte aligned reference frame buffer addresses for the motion compensation operation in AVC/VC1/MPEG2. AVC can specify up to 16 YUV frame-based surfaces for both forward and backward references, i.e. L0+L1 total = 16 max. Any entry can be assigned to L0 or L1 or both lists. But VC1 and MPEG2, worst case, can use up to 2 YUV frame-based surfaces for both forward and backward references:P-MB: RefAddr[0] - temporal closest previous field of a reference frame (can be the current frame)RefAddr[1] - next temporal closest previous field of a reference frame (must be different from the current frame)It is a variant (without the LongTermRefPic specification) of the RefFrameList[16] defined in AVC DXVA Spec. RefAddr[0-15] is indexed by frame_storeID >>1. It is not a packed list, i.e. invalid entries can scatter among the list. All invalid addresses must be set to a valid address RefAddr[0] by the driver. The same applies to VC1 and MPEG2.
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>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.</td></tr>
</table></td></tr>
<tr><td></td><td>5:4</td><td><b>Reference Picture (RefAddr[0-15]) - Arbitration Priority Control</b>
This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.</td></tr>
</table>

<table>
<tr><th colspan="3">MFX_PIPE_BUF_ADDR_STATE</th></tr>
<tr><td></td><td></td><td><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><b>Reserved</b></td></tr>
<tr><td></td><td>2</td><td><b>Reference Picture (RefAddr[0-15]) - Graphics Data Type (GFDT)</b>
This field contains the GFDT bit for this surface when writes occur. GFDT can also be set by the GTT. The effective GFDT is the logical OR of this field with the GFDT from the GTT entry. This field is ignored for reads. H/W only reads this bit from the very first RefAddr[0][bit 3:0], all other RefAddr[i][bit 3:0] are ignored by H/W and are assumed to have the same values as that of RefAddr[0].</td></tr>
<tr><td></td><td>1:0</td><td><b>Reference Picture (RefAddr[0-15]) - Cacheability Control</b>
This field controls cacheability. H/W only reads this bit from the very first RefAddr[0][bit 3:0], all other RefAddr[i][bit 3:0] are ignored by H/W and are assumed to have the same values as that of RefAddr[0].<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 cached</td><td>data is not cached</td></tr><tr><td>11b</td><td>Cached</td><td>Data is cached</td></tr></table></td></tr>
<tr><td>23</td><td>31:6</td><td><b>Macroblock Status Buffer Base Address (MacroblockStatAddr)</b><table><tr><td>Format:</td><td>GraphicsAddress[31:6]</td></tr></table>Specifies the 64 byte aligned address for reading the per-MB indirect data from memory when MacroblockStatEnable is set in the MFX_AVC_IMG_STATE Command. For decoder: this field is ignored by hardware. For encoder: this field is used for dynamic repeat of frame in PAK for Rate Control. Also used for feeding coding information back to the Host, Video Preprocessing Unit and ENC Unit.All data are written in fixed formats, and therefore all record sizes are known in the hardware. Hardware can calculate the offset into this base address for per-MB data.</td></tr>
<tr><td></td><td>5:4</td><td><b>Arbitration Priority Control</b>
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><b>Reserved</b></td></tr>
<tr><td></td><td>2</td><td><b>Graphics Data Type (GFDT</b>
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>
</table>

MFX_PIPE_BUF_ADDR_STATE

<table>
<tr><td></td><td>1:0</td><td><b>Cacheability Control</b>
This field controls cacheability.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>00b</td><td>GTT</td><td>use cacheability control bits from GTT entry</td></tr><tr><td>01b</td><td>Not cached</td><td>data is not cached</td></tr><tr><td>11b</td><td>Cached</td><td>Data is cached</td></tr></table></td></tr>
<tr><td>24</td><td>31:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
</table>

MFX_PIPE_MODE_SELECT

Source:	VideoCS
Length Bias:	2

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_COMMON Format: OpCode
	26:24	**Opcode** Default Value: 0h MFX_COMMON_STATE Format: OpCode
	23:21	**SubOpA** Default Value: 0h Format: OpCode
	20:16	**SubOpB** Default Value: 0h MFX_PIPE_MODE_SELECT Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Total Length - 2 Value | Name | Description 3h | DWORD_COUNT_n **[Default]** | Excludes DWord (0,1)
1	31	**Reserved**
	30	**Reserved**
	29	**Reserved**
	28:27	**Reserved**
	26:25	**Reserved** Format: MBZ
	24	**Reserved**

MFX_PIPE_MODE_SELECT

23:19 **Reserved**

Format:	MBZ

17 **Decoder Short Format Mode**

For IT mode, this bit must be 0.

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

16:15 **Decoder Mode select**

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

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

14:13 **Reserved**

Format:	MBZ

12 **Reserved**

Format:	MBZ

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

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

Value	Name
0h	Disable
1h	Enable

10 **Stream-Out Enable**

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

Value	Name
0h	Disable
1h	Enable

Programming Notes
In decoder modes: The Stream-Out feature is added to support transcoding. While decoding

MFX_PIPE_MODE_SELECT

<table>
<tr><td></td><td></td><td>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.</td></tr>
<tr><td></td><td>9</td><td><b>Post Deblocking Output Enable (PostDeblockOutEnable)</b>
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.
<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>
<tr><td></td><td>8</td><td><b>Pre Deblocking Output Enable (PreDeblockOutEnable)</b>
This field controls the output write for the reconstructed pixels BEFORE the deblocking filter.
<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>
<tr><td></td><td>7:6</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>5</td><td><b>Stitch Mode</b>
<table>
<tr><td>Exists If:</td><td>//CodecSel=Encode and StandardSel=AVC</td></tr>
</table>
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Not in stitch mode</td><td></td></tr>
<tr><td>1h</td><td>In the special stitch mode</td><td>This mode can be used for any Codec as long as bitfield conditions are met.</td></tr>
</table></td></tr>
<tr><td></td><td>4</td><td><b>Codec Select</b>
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Decode</td><td></td></tr>
<tr><td>1h</td><td>Encode</td><td>Valid only if StandardSel is AVC, MPEG2)</td></tr>
</table></td></tr>
<tr><td></td><td>3:0</td><td><b>Standard Select</b>
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0000b</td><td>MPEG2</td><td></td></tr>
<tr><td>0001b</td><td>VC1</td><td></td></tr>
<tr><td>0010b</td><td>AVC</td><td>Covers both AVC and MVC</td></tr>
<tr><td>0011b</td><td>JPEG</td><td></td></tr>
<tr><td>0110b</td><td>Reserved</td><td></td></tr>
<tr><td>0111b</td><td>Reserved</td><td></td></tr>
</table></td></tr>
<tr><td>2</td><td>31</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
</table>

MFX_PIPE_MODE_SELECT

<table>
<tr><td></td><td>30</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>29</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>28:26</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>25</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>24</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>23</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>22:21</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>20</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>19</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>18</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>17</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>16</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>15</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>14</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>13</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>12</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>10</td><td><b>MPC pref08x8_disable Flag (Default 0)</b><table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>Disable</td></tr><tr><td>1</td><td>Enable</td></tr></table></td></tr>
<tr><td></td><td>9</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7</td><td><b>Reserved</b></td></tr>
</table>

<table>
<tr><th colspan="3">MFX_PIPE_MODE_SELECT</th></tr>
<tr><td></td><td>6</td><td><b>Clock gate Enable at Slice-level</b>
BitFieldDesc:
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>Disable</td><td>Disable Slice-level Clock gating, Unit-level Clock gating will apply</td></tr>
<tr><td>1</td><td>Enable</td><td>Enable Slice-level Clock gating, overrides any Unit level Clock gating</td></tr>
</table>
</td></tr>
<tr><td></td><td>5</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>4</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>3</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>2</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>1:0</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table>
</td></tr>
<tr><td>3</td><td>31:0</td><td><b>Pic Status/Error Report ID</b>
<table>
<tr><td>Exists If:</td><td>//Decoder Mode Only</td></tr>
<tr><td>Format:</td><td>U32</td></tr>
</table>
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.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>32-bit unsigned</td><td>Unique ID Number</td></tr>
<tr><td>1h</td><td>Reserved</td><td></td></tr>
</table>
</td></tr>
<tr><td>4</td><td>31:0</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table>
</td></tr>
</table>

MFX_QM_STATE

Source:	VideoCS
Length Bias:	2

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

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

<table>
<tr><th colspan="3">MFX_QM_STATE</th></tr>
<tr><td></td><td>1:0</td><td><b>AVC</b>
<table><tr><td>Exists If:</td><td>//AVC- Decoder Only</td></tr></table>
<b>For AVC QM Type</b>: This field specifies which Quantizer Matrix is loaded.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0</td><td>AVC_4x4_Intra_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs)</td></tr>
<tr><td>1</td><td>AVC_4x4_Inter_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs)</td></tr>
<tr><td>2</td><td>AVC_8x8_Intra_MATRIX</td></tr>
<tr><td>3</td><td>AVC_8x8_Inter_MATRIX</td></tr>
</table></td></tr>
<tr><td>2..33</td><td>31:0</td><td><b>Forward Quantizer Matrix</b>
<table><tr><td>Format:</td><td>U32</td></tr></table>
The format of a Quantizer Matrix is an 8x8 matrix in raster order. Each element is an unsigned byte.</td></tr>
</table>

MFX_STATE_POINTER

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**	
		Format:	MBZ
	11:0	**DWord Length**	
		Default Value:	0h DWORD_COUNT_n

MFX_STATE_POINTER

<table>
<tr><td></td><td></td><td><table><tr><td>Format:</td><td>=n Total Length - 2</td></tr></table></td></tr>
<tr><td>1</td><td>31:5</td><td><b>State Pointer</b><table><tr><td>Format:</td><td>GeneralStateOffset[31:5]Indirect State Buffer</td></tr></table>Specifies the 32-byte aligned address of an Indirect State Buffer. This pointer is relative to the General State Base Address.</td></tr>
<tr><td></td><td>4:2</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>1:0</td><td><b>State Pointer Index</b>
Specifies one of the four indirect state pointers to program.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>00b</td><td></td><td>indirect state pointer 0 (image state)</td></tr><tr><td>01b</td><td></td><td>indirect state pointer 1 (slice state)sc</td></tr><tr><td>10b</td><td></td><td>indirect state pointer 2</td></tr><tr><td>11b</td><td></td><td>indirect state pointer 3</td></tr></table></td></tr>
</table>

MFX_STITCH_OBJECT

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.

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** Format: MBZ
	11:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) = Variable Length in DW (&gt;= 3) Format: =n Total Length - 2 If it is 3, it indicates the absent of inline data.
1	31:18	**Reserved** Format: MBZ
	17:16	**Source Data Starting Byte Offset** Source Data Starting Byte Position within the very first inline DW.
	15:14	**Reserved**

MFX_STITCH_OBJECT

		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** **Name** [1,32]
	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** 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

Source:	VideoCS
Length Bias:	2

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

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

Since there is only one media surface state being active during the entire encoding/decoding process, all the uncompressed/reconstructed pictures are defined to have the same surface state. The primary difference among picture surface states is their individual programmed base addresses, which are provided by other state commands and not included in this command. MFX engine is making the association of surface states and corresponding buffer base addresses.
MFX engine currently supports only one media surface type for video and that is the NV12 (Planar YUV420 with interleaved U (Cb) and V (Cr). For optimizing memory efficiency based on access patterns, only TileY is supported. For JPEG decoder, only IMC1 and IMC3 are supported. Pitch can be wider than the Picture Width in pixels and garbage will be there at the end of each line. The following describes all the different formats that are supported and not supported in MFX :

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

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

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

This surface state here is identical to the Surface State for deinterlace and sample_8x8 messages described in the Shared Function Volume and Sampler Chapter.
For non pixel data, such as row stores, indirect data (Compressed Slice Data, AVC MV record, Coeff record and AVC ILDB record) and streamin/out and output compressed bitstream, a linear buffer is employed. For row stores, the H/W is designed to guarantee legal memory accesses (read and write). For the remaining cases, indirect object base address, indirect object address upper bound, object data start address (offset) and object data length are used to fully specified their corresponding buffer. This

MFX_SURFACE_STATE

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 decoder pipeline and kernel to perform this function, it is going to be done in the video post-processing scaler or display controller scale as a separate step and controller.
All video codec surfaces must be NV12 Compliant, except JPEG. U/V vertical must be MB aligned for all video codec (further contrained for field picture), but JPEG can be block aligned. All video codec and JPEG uses Tiled - Y format only, for uncompressed pixel surfaces.
Even for JPEG planar 420 surface, application may provide only 1 buffers, but there is still only one single surface state for all of them. If IMC equal to 1, 2, 3 or 4, U and V have the pitch same as Y. And U and V will have different offset, each offset is block aligned.

DWord	Bit	Description	
0	31:29	**Command Type**	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	**Pipeline**	
		Default Value:	2h MFX_COMMON
		Format:	OpCode
	26:24	**Opcode**	
		Default Value:	0h MFX_COMMON_STATE
		Format:	OpCode
	23:21	**SubOpA**	
		Default Value:	0h
		Format:	OpCode
	20:16	**SubOpB**	
		Default Value:	1h
		Format:	OpCode
	15:12	**Reserved**	
		Format:	MBZ
	11:0	**DWord Length**	
		Format:	=n Total Length - 2

Value	Name	Description
4h	DWORD_COUNT_n **[Default]**	Excludes DWord (0,1)

MFX_SURFACE_STATE

1 | 31:4 | **Reserved**

Format:	MBZ

Surface Base Address is NOT used for codec H/W. This filed is reserved for 3D surface state compatibility. MFX pipeline gets this address from MFX_PIPE_BUF_ADDR_STATE for different buffers.

3:0 | **Reserved**

Format:	MBZ

2 | 31:18 | **Height**

Format:	U14-1 Height

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

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

Programming Notes

- For AVC: For frame picture is a multiple of 16; for field picture is a multiple of 32
- For VC1: For progressive frames, the frame height and frame width is a multiple of 2 pixels. For interlaced frames, the frame height shall be a multiple of 4 pixels, and its width is a multiple of 2 pixels, based on a PLANAR_420 surface.

Video Codecs must program less than and equal to 4K.

17:4 | **Width**

Format:	U14-1 Width

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

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

Programming Notes

- The Width specified by this field multiplied by the pixel size in bytes must be less than or equal to the surface pitch (specified in bytes via the Surface Pitch field).
- Width (field value + 1) must be a multiple of 2 for PLANAR_420,
- MFX HW does not use this field, the picture width is read from IMG State instead, because this field may not equal to the actual picture width. This field is used by the KMD to allocate surface in GTT.

Video Codecs must program less than and equal to 4K.

<table>
<tr><th colspan="3">MFX_SURFACE_STATE</th></tr>
<tr><td></td><td>3:2</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>1:0</td><td><b>Cr(V)/Cb(U) Pixel Offset V Direction</b><table><tr><td>Format:</td><td>U0.2 exactly as shown in the original spec</td></tr></table>Specifies the distance to the U/V values with respect to the even numbered Y channels in the V direction<table><tr><th>Programming Notes</th></tr><tr><td>This field is ignored for all formats except PLANAR_420_8</td></tr></table></td></tr>
<tr><td>3</td><td>31:28</td><td><b>Surface Format</b>
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.<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>YCRCB_NORMAL</td><td></td></tr>
<tr><td>1</td><td>YCRCB_SWAPUVY</td><td></td></tr>
<tr><td>2</td><td>YCRCB_SWAPUV</td><td></td></tr>
<tr><td>3</td><td>YCRCB_SWAPY</td><td></td></tr>
<tr><td>4</td><td>PLANAR_420_8</td><td>(NV12, IMC1,2,3,4, YV12)</td></tr>
<tr><td>5</td><td>PLANAR_411_8</td><td>Deinterlace Only</td></tr>
<tr><td>6</td><td>PLANAR_422_8</td><td>Deinterlace Only</td></tr>
<tr><td>7</td><td>STMM_DN_STATISTICS</td><td>Deinterlace Only</td></tr>
<tr><td>8</td><td>R10G10B10A2_UNORM</td><td>Sample_8x8 Only</td></tr>
<tr><td>9</td><td>R8G8B8A8_UNORM</td><td>Sample_8x8 Only</td></tr>
<tr><td>10</td><td>R8B8_UNORM (CrCb</td><td>Sample_8x8 Only</td></tr>
<tr><td>11</td><td>R8_UNORM (Cr/Cb)</td><td>Sample_8x8 Only</td></tr>
<tr><td>12</td><td>Y8_UNORM</td><td>Sample_8x8 Only</td></tr>
<tr><td>13,15</td><td>Reserved</td><td></td></tr>
</table></td></tr>
<tr><td></td><td>27</td><td><b>Interleave Chroma</b><table><tr><td>Format:</td><td>Enable</td></tr></table>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.)<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>1</td><td>Enable</td></tr>
<tr><td>0</td><td>Disable</td></tr>
</table></td></tr>
</table>

<table>
<tr><th colspan="3">MFX_SURFACE_STATE</th></tr>
<tr><td></td><td>26</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>25:22</td><td><b>Surface Object Control State (MEMORY_OBJECT_CONTROL_STATE)</b>
<table><tr><td>This 4-bit field is used in various state commands and indirect state objects.</td></tr></table>
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>3</td><td>Reserved</td><td></td></tr>
<tr><td>2</td><td>Graphics Data Type (GFDT)</td><td>This field contains the GFDT bit for this surface when writes occur. GFDT can also be set by the GTT. The effective GFDT is the logical OR of this field with the GFDT from the GTT entry. This field is ignored for reads.
Format = U1</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>This field is ignored; H/W uses those values programmed in each of the Buf Address State entries instead.</td></tr>
</table></td></tr>
<tr><td></td><td>21:20</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>19:3</td><td><b>Surface Pitch</b>
<table><tr><td>Format:</td><td>U17-1 pitch in Bytes</td></tr></table>
This field specifies the surface pitch in (#Bytes).
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>[0,2047]</td><td></td><td>to [1B, 2048B]</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>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]</td></tr>
</table></td></tr>
<tr><td></td><td>2</td><td><b>Half Pitch for Chroma</b>
<table><tr><td>Format:</td><td>Enable</td></tr></table>
(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)</td></tr>
<tr><td></td><td>1</td><td><b>Tiled Surface</b>
<table><tr><td>Format:</td><td>Boolean</td></tr></table>
(This field must be set to TRUE: Tiled)This field specifies whether the surface is tiled. This field is ignored by MFX
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>False</td><td>Linear</td></tr>
<tr><td>1</td><td>True</td><td>Tiled</td></tr>
</table></td></tr>
</table>

<table>
<tr><th colspan="3">MFX_SURFACE_STATE</th></tr>
<tr><td></td><td></td><td><table><tr><th>Programming Notes</th></tr><tr><td>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.</td></tr></table></td></tr>
<tr><td></td><td>0</td><td><b>Tile Walk</b><table><tr><td>Format:</td><td>3D_Tilewalk</td></tr></table>(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.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>XMAJOR</td><td>TILEWALK_XMAJOR</td></tr><tr><td>1h</td><td>YMAJOR</td><td>TILEWALK_YMAJOR</td></tr></table><table><tr><th>Programming Notes</th></tr><tr><td>The corresponding cache(s) must be invalidated before a previously accessed surface is accessed again with an altered state of this bit</td></tr></table></td></tr>
<tr><td>4</td><td>31</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>30:16</td><td><b>X Offset for U(Cb)</b><table><tr><td>Format:</td><td>U15 Pixel Offset</td></tr></table>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)<table><tr><th>Programming Notes</th></tr><tr><td>For PLANAR_420 and PLANAR_422 surface formats, this field must be zero.</td></tr></table></td></tr>
<tr><td></td><td>15</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>14:0</td><td><b>Y Offset for U(Cb)</b><table><tr><td>Format:</td><td>U15 Pixel Row Offset</td></tr></table>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.<table><tr><th>Programming Notes</th></tr><tr><td>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.</td></tr></table></td></tr>
<tr><td>5</td><td>31:29</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
</table>

MFX_SURFACE_STATE

	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

Source:	VideoCS
Length Bias:	2

This is a picture level command and should be issued only once, even for a multi-slices picture. This command is only valid in the VC1 decoding in VLD modes. 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 Bytes per MB, to store the top and bottom field MVs (32-bit MVx,y each). Note that if there is a I picture before a B picture the DmvSurfaceValid state in MFX_VC1_PIC_STATE Command will NOT be set and zero's DMV's will be assumed while decoding the B picture. That is, there is no explicit DMV buffer for an I-picture.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h PARALLEL_VIDEO_PIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Pipeline</b><table><tr><td>Default Value:</td><td>2h MFX_VC1_DIRECTMODE_STATE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>26:24</td><td><b>Media Command Opcode</b><table><tr><td>Default Value:</td><td>2h VC1</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>23:21</td><td><b>SubOpcode A</b><table><tr><td>Default Value:</td><td>0h Common</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>20:16</td><td><b>SubOpcode B</b><table><tr><td>Default Value:</td><td>2h MEDIA_</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:12</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>0001h Excludes DWord (0,1)</td></tr><tr><td>Format:</td><td>=n Total Length - 2</td></tr></table></td></tr>
<tr><td>1</td><td>31:6</td><td><b>Direct MV Write Buffer Base Address for the Current Picture</b>
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</td></tr>
<tr><td></td><td>5:4</td><td><b>Direct MV Write Buffer Base Address - Arbitration Priority Control</b><table><tr><td>Format:</td><td>U2 Enumerated Type</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></table></td></tr>
</table>

MFX_VC1_DIRECTMODE_STATE

<table>
<tr><td></td><td></td><td><table><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><b>Reserved</b></td></tr>
<tr><td></td><td>2</td><td><b>Direct MV Write Buffer Base Address - Graphics Data Type (GFDT) for the Current Picture</b><table><tr><td>Default Value:</td><td>0h</td></tr><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.</td></tr>
<tr><td></td><td>1:0</td><td><b>Direct MV Write Buffer Base Address - Cacheability Control for the Current Picture</b><table><tr><td>Format:</td><td>U2 Enumerated Type</td></tr></table>This field controls cacheability.<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>use cacheability control bits from GTT entry</td></tr><tr><td>01b</td><td>data is not cached</td></tr><tr><td>11b</td><td>data is cached</td></tr></table></td></tr>
<tr><td>2</td><td>31:6</td><td><b>Direct MV Read Buffer Base Address for the Reference Picture</b>
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></td><td>5:4</td><td><b>Direct MV Read Buffer - Arbitration Priority Control</b><table><tr><td>Format:</td><td>U2 Enumerated Type</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></td><td>3</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>2</td><td><b>Direct MV Read Buffer - Graphics Data Type (GFDT) for the Reference Picture</b><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.\</td></tr>
<tr><td></td><td>1:0</td><td><b>Direct MV Read Buffer - Cacheability Control for the Reference Picture</b><table><tr><td>Format:</td><td>U2 Enumerated Type</td></tr></table></td></tr>
</table>

<table>
<tr><th colspan="3">MFX_VC1_DIRECTMODE_STATE</th></tr>
<tr><td></td><td></td><td>This field controls cacheability.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>00b</td><td>use cacheability control bits from GTT entry</td></tr>
<tr><td>01b</td><td>data is not cached</td></tr>
<tr><td>11b</td><td>data is cached</td></tr>
</table>
</td></tr>
</table>

MFX_VC1_PRED_PIPE_STATE

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** Format: MBZ
	11:0	**DWord Length** Default Value: 0004h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:16	**Reserved** Format: MBZ
	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

MFX_VC1_PRED_PIPE_STATE

		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** Format: MBZ
	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** Format: MBZ
	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** Format: MBZ
	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

MFX_VC1_PRED_PIPE_STATE		
		wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	15:14	**Reserved** Format: MBZ
	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**

MFX_VC1_PRED_PIPE_STATE

		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 - 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_WAIT

Source:	VideoCS
Length Bias:	1

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

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 03h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Command Subtype** Default Value: 01h MFX_SINGLE_DW Format: OpCode
	26:16	**Sub-Opcode** Default Value: 0h MFX_WAIT Format: OpCode
	15:10	**Reserved** 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** Format: MBZ
	5:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2

MI_ARB_CHECK

Source:	BlitterCS
Length Bias:	1

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.

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

Source:	RenderCS
Length Bias:	1

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

- 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.

This instruction can be in either a ring buffer or batch buffer.

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

Source:	VideoCS
Length Bias:	1

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.

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

Source:	RenderCS
Length Bias:	1

The MI_ARB_ON_OFF instruction is used to mask/differ the below asynchronous events when arbitration is disabled: • PSMI Context Switch Request • Sync Flush • Power FLush Block This command should always be used as an off-on pair around the sequence of instructions to be protected from above mentioned asynchronous events.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><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><b>MI Command Opcode</b><table><tr><td>Default Value:</td><td>08h MI_ARB_ON_OFF</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>22:1</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>0</td><td><b>Arbitration Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>This field enables or disables arbitration in HW.<table><tr><th>Value</th><th>Name</th></tr><tr><td>0h</td><td>Disabled</td></tr><tr><td>1h</td><td>Enabled</td></tr></table></td></tr>
</table>

MI_ARB_ON_OFF

Source:	VideoCS
Length Bias:	1

The MI_ARB_ON_OFF instruction is used to mask/differ the below asynchronous events when arbitration is disabled: • PSMI Context Switch Request • Sync Flush • Power FLush Block This command should always be used as an off-on pair around the sequence of instructions to be protected from above mentioned asynchronous events.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND
	28:23	**MI Command Opcode** Default Value: 08h MI_ARB_ON_OFF
	22:1	**Reserved** Format: MBZ
	0	**Arbitration Enable** Format: Enable This field enables or disables arbitration in HW. **Value** | **Name** 0h | Disabled 1h | Enabled

MI_BATCH_BUFFER_END

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** Format: MBZ

MI_BATCH_BUFFER_END

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

Source:	VideoCS
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_START

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
• 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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><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><b>MI Command Opcode</b><table><tr><td>Default Value:</td><td>31h MI_BATCH_BUFFER_START</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>22</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>21:9</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>8</td><td><b>Address Space Indicator</b><table><tr><td>Format:</td><td>MI_BufferSecurityType</td></tr></table><table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>GGTT</td><td>This batch buffer is secure and will be accessed via the GGTT.</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><b>DWord Length</b><table><tr><td>Format:</td><td>=n</td></tr></table>Total - Bias<table><tr><th>Value</th><th>Name</th></tr><tr><td>0h</td><td>Excludes DWord (0,1) <b>[Default]</b></td></tr></table></td></tr>
<tr><td>1</td><td>31:2</td><td><b>Batch Buffer Start Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:2]BatchBuffer</td></tr></table>This field specifies Bits 31:2 of the starting address of the batch buffer.</td></tr>
</table>

MI_BATCH_BUFFER_START		
	1:0	**Reserved** Format: MBZ

MI_BATCH_BUFFER_START

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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b>
<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><b>MI Command Opcode</b>
<table>
<tr><td>Default Value:</td><td>31h MI_BATCH_BUFFER_START</td></tr>
<tr><td>Format:</td><td>OpCode</td></tr>
</table></td></tr>
<tr><td></td><td>22</td><td><b>2nd Level Batch Buffer</b>
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.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>1st level batch</td><td>Place the batch buffer address in the 1st (traditional) level batch address storage element</td></tr>
<tr><td>1h</td><td>2nd level batch</td><td>Place the batch buffer address in the 2nd level batch address storage element</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td><ul><li>2nd level batch buffer chaining is not supported.</li></ul></td></tr>
</table></td></tr>
<tr><td></td><td>21:10</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>9</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>8</td><td><b>Address Space Indicator</b>
<table>
<tr><th>Description</th></tr>
<tr><td>This field must be 0 unless the Per-Process GTT Enable is 1.</td></tr>
</table>
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>GGTT space</td><td>This batch buffer will be accessed via the GGTT.</td></tr>
<tr><td>1</td><td>PPGTT</td><td>This batch buffer will be accessed via the PPGTT.</td></tr>
</table></td></tr>
</table>

MI_BATCH_BUFFER_START

<table>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Format:</td><td>=n Total Length - 2</td></tr></table><table><tr><th>Value</th><th>Name</th></tr><tr><td>0h</td><td>Excludes DWord (0,1) <b>[Default]</b></td></tr></table></td></tr>
<tr><td>1</td><td>31:2</td><td><b>Batch Buffer Start Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:2]</td></tr></table><table><tr><th>Programming Notes</th></tr><tr><td><ul><li>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.</li><li>The selection of PPGTT vs. GGTT for the batch buffer is determined by the Buffer Security Indicator (bit8).</li></ul></td></tr></table></td></tr>
<tr><td></td><td>1:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
</table>

MI_BATCH_BUFFER_START

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.

Programming Notes

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. Prior to sending batch buffer start command with clear command buffer enable set, software has to ensure pipe is flushed explicitly by sending MI_FLUSH.

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	**Reserved** Format: MBZ
	21:17	**Reserved** Format: MBZ
	16	**Reserved** Format: MBZ
	15	**Reserved** Format: MBZ
	14	**Reserved** Format: MBZ
	13	**Reserved** Format: MBZ
	12	**Reserved**
	11	**Clear Command Buffer Enable** Format: Enable This batch buffer needs to be preceded by a MI_FLUSH command or PIPE_CONTROL with CS Stall set.
	10	**Reserved** Format: MBZ
	9	**Reserved**

MI_BATCH_BUFFER_START

<table>
<tr><td></td><td></td><td><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>8</td><td><b>Address Space Indicator</b>
<table>
<tr><th>Description</th></tr>
<tr><td>SW must ensure the "Address Space Indicator" of the chained batch buffer to be same as the initial batch buffer. Ex: If the MI_BATCH_BUFFER_START executed from Ring Buffer has "Address Space Indicator" as "PPGTT" then all subsequent chained batch buffers (not second level Batch Buffers) must be in "PPGTT".
Not complying to above programming will result in unknown behavior of HW.</td></tr>
<tr><td>This field must be '0' unless the Per-Process GTT Enable is '1'</td></tr>
</table>
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>GGTT</td><td>This batch buffer will be accessed via the GGTT.</td></tr>
<tr><td>1h</td><td>PPGTT</td><td>This batch buffer will be accessed via the PPGTT.</td></tr>
</table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b>
<table>
<tr><td>Default Value:</td><td>0h Excludes DWord (0,1)</td></tr>
<tr><td>Format:</td><td>=n Total - Bias</td></tr>
</table></td></tr>
<tr><td>1</td><td>31:2</td><td><b>Batch Buffer Start Address</b>
<table>
<tr><td>Format:</td><td>GraphicsAddress[31:2]BatchBuffer</td></tr>
</table>
This field specifies Bits 31:2 of the starting address of the batch buffer.</td></tr>
<tr><td></td><td>1:0</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
</table>

MI_CLFLUSH

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. This command is not privileged.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><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><b>MI Command Opcode</b><table><tr><td>Default Value:</td><td>27h Store DW MI_CLFLUSH</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>22</td><td><b>Use Global GTT</b>
This bit must be 1 if the <b>Per Process GTT Enable</b> 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>This command will use the global PPGTT to translate the Address.</td></tr><tr><td>1h</td><td>Global Graphics Address</td><td>This command will use the global GTT to translate the Address.</td></tr></table></td></tr>
<tr><td></td><td>21:10</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>9:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>1h</td></tr><tr><td>Format:</td><td>=n Total Length - 2. Excludes DWord (0,1).</td></tr></table></td></tr>
<tr><td>1</td><td>31:12</td><td><b>Page Base Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:12]</td></tr></table>4KB aligned Page Address which software requires hardware to flush to DRAM.</td></tr>
<tr><td></td><td>11:6</td><td><b>Starting Cacheline Offset</b><table><tr><td>Format:</td><td>U6 Zero based starting cacheline offset to the Page Base Address.</td></tr></table></td></tr>
<tr><td></td><td>5:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>2..n</td><td>31:0</td><td><b>DW Representing a Half Cache Line</b><table><tr><td>Format:</td><td>MBZ</td></tr></table>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.<table><tr><th>Programming Notes</th></tr><tr><td>Always even number of "DW Representing 1/2 cacheline" terms must be programmed.</td></tr></table></td></tr>
</table>

MI_CONDITIONAL_BATCH_BUFFER_END

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 If set, this command will use the global GTT to translate the Compare Address. If clear, the PPGTT will be used to translate the Compare Address. This bit must be 1 if the Per Process GTT Enable bit is clear.
	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** Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2 Value: 0h — Name: Excludes DWord (0,1) **[Default]**
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

		MI_CLFLUSH
		than this dword, the execution of the command buffer should continue.
2	31:3	**Compare Address** Format: GraphicsAddress[31:3] 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** Format: MBZ

MI_CONDITIONAL_BATCH_BUFFER_END

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**. 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, no comparison takes place.
	20	**Reserved**
	19:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2. Excludes DWord (0,1). Value: 0h — Name: **[Default]**
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.
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**

MI_CONDITIONAL_BATCH_BUFFER_END			
		Format:	MBZ

MI_DISPLAY_FLIP

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.

The full packet must be contained within the same cache line.

There must be at least one valid command following this packet.

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

MI_DISPLAY_FLIP

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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b>
<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><b>MI Command Opcode</b>
<table><tr><td>Default Value:</td><td>14h MI_DISPLAY_FLIP</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>22</td><td><b>Async Flip Indicator</b>
<table><tr><td>Format:</td><td>Enable</td></tr></table>
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.</td></tr>
<tr><td></td><td>21:19</td><td><b>Display (Plane) Select</b>
This field selects which display plane is to perform the flip operation.
<table><tr><th>Value</th><th>Name</th></tr>
<tr><td>0h</td><td>Display Plane A</td></tr>
<tr><td>1h</td><td>Display Plane B</td></tr>
<tr><td>2h</td><td>Display Sprite A</td></tr>
<tr><td>3h</td><td>Display Sprite B</td></tr>
<tr><td>4h</td><td>Display Plane C</td></tr>
<tr><td>5h</td><td>Display Sprite C</td></tr></table></td></tr>
<tr><td></td><td>18:17</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>16</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>15:13</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>12:8</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b>
<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></tr></table></td></tr>
</table>

<table>
<tr><th colspan="3">MI_DISPLAY_FLIP</th></tr>
<tr><td></td><td></td><td><table><tr><td>0h</td><td>Excludes DWord (0,1) <b>[Default]</b></td></tr><tr><td>1h</td><td>Reserved</td></tr></table></td></tr>
<tr><td>1</td><td>31</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>30:16</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>15:6</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>5:1</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>0</td><td><b>Tile Parameter</b><table><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 <b>[Default]</b></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><b>Display Buffer Base Address</b><table><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><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>2</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>1:0</td><td><b>Flip Type</b>
This field specifies whether the flip operation should be performed asynchronously to vertical retrace.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>00b</td><td>Sync Flip <b>[Default]</b></td><td>The flip will occur during the vertical blanking interval - thus avoiding any tearing artifacts.</td></tr><tr><td>01b</td><td>Async Flip</td><td>The flip will occur "as soon as possible" - and may exhibit tearing artifacts</td></tr><tr><td>1b</td><td>Reserved</td><td></td></tr></table></td></tr>
</table>

MI_FLUSH

Source:	RenderCS
Length Bias:	1

Description
The MI_FLUSH 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 and the read caches are invalidated including the texture cache accessed via the Sampler or the data port. In addition, this command can also be used to: • Flush any dirty data in the Render Cache to memory. This is done by default, however this can be inhibited. • Invalidate the state and command cache. **Usage Note:** After this command is completed and followed by a Store DWord-type command, CPU access to graphics memory will be coherent (assuming the Render Cache flush is not inhibited). This command is specific to the render engine. Other engines use MI_FLUSH_DW. To use this command, bit 12 in the MI_MODE(0x209c) must be enabled.
If GFX_MODE (0x229C) bit 13, this command will cause a config write to MMIO register space with the address 0x4f100.
MI_FLUSH command is no longer validated or supported. Use at your own risk.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 04h MI_FLUSH Format: OpCode
	22:7	**Reserved** Format: MBZ
	6	**Reserved**
	5	**Indirect State Pointers Disable** Format: Disable At the completion of the flush, the indirect state pointers in the hardware will be considered as invalid. I.e., the indirect pointers will not be restored for the context.
	4	**Generic Media State Clear** Format: Disable If set, all generic media state context information will not be included with the next context save, assuming no new state is initiated after the flush. If clear, the generic media state context save state will not be affected. An MI_FLUSH with this bit set should be issued once all the Media Objects that will be processed by a given persistent root

<table>
<tr><th colspan="3">MI_DISPLAY_FLIP</th></tr>
<tr><td rowspan="5"></td><td></td><td>thread have been issued or when an MI_SET_CONTEXT switching from a generic media context to a 3D context completes. When using MI_SET_CONTEXT, once state is programmed, it will be saved and restarted as part of any context each time that context is saved/restored until an MI_FLUSH with this bit set is issued in that context.</td></tr>
<tr><td>3</td><td><b>Reserved</b></td></tr>
<tr><td>2</td><td><b>Render Cache Flush Inhibit</b>
<table><tr><td>Format:</td><td>Boolean</td></tr></table>
If set, the Render Cache is not flushed as part of the processing of this command.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Flush</td><td>Flush the Render Cache.</td></tr>
<tr><td>1h</td><td>Don't Flush</td><td>Do not flush the Render Cache.</td></tr>
</table></td></tr>
<tr><td>1</td><td><b>State/Instruction Cache Invalidate</b>
<table><tr><td>Format:</td><td>Boolean</td></tr></table>
If set, Invalidates the State and Instruction Cache.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Don't Invalidate</td><td>Leave State/Instruction Cache unaffected.</td></tr>
<tr><td>1h</td><td>Invalidate</td><td>Invalidate State/Instruction Cache.</td></tr>
</table></td></tr>
<tr><td>0</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
</table>

MI_FLUSH_DW

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** Format: U1
	21	**Store Data Index** Format: U1 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 hardware status page.
	20:19	**Reserved** Format: MBZ
	18	**TLB Invalidate** Format: U1 **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	**Synchronize GFDT surface** Format: U1 If enabled, at the end of the current flush the last level cache is cleared of all the cachelines which have been marked with the special GFDT flags. Store DW must be enabled
	16	**Reserved** Format: MBZ
	15:14	**Post-Sync Operation** BitFieldDesc

MI_FLUSH_DW

<table>
<tr><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>No Write</td><td>No write occurs as a result of this instruction. This can be used to implement a "trap" operation, etc.</td></tr>
<tr><td>1h</td><td>Write Immediate Data QWord</td><td>Write the QWord containing Immediate Data Low, High DWs to the Destination Address</td></tr>
<tr><td>2h</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>3h</td><td></td><td>Write the TIMESTAMP register to the Destination Address with a granularity of 80ns.
The upper 28 bits of the TIMESTAMP register are tied to '0'.</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>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.</td></tr>
</table>
</td></tr>
<tr><td></td><td>13:10</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table>
</td></tr>
<tr><td></td><td>9</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table>
</td></tr>
<tr><td></td><td>8</td><td><b>Notify Enable</b>
<table>
<tr><td>Format:</td><td>U1</td></tr>
</table>
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.
</td></tr>
<tr><td></td><td>7:6</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table>
</td></tr>
<tr><td></td><td>5:0</td><td><b>DWord Length</b>
<table>
<tr><td>Format:</td><td>=n Total Length - 2</td></tr>
</table>
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>2h</td><td>Excludes DWord (0,1) = 1 for DWord, 2 for QWord <b>[Default]</b></td></tr>
</table>
</td></tr>
<tr><td>1</td><td>31:3</td><td><b>Address</b>
<table>
<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><b>Destination Address Type</b>
Defines address space of Destination Address
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>PPGTT</td><td>Use PPGTT address space for DW write</td></tr>
</table>
</td></tr>
</table>

MI_FLUSH_DW

<table>
<tr><td></td><td></td><td><table><tr><td>1h</td><td>GGTT</td><td>Use GGTT address space for DW write</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><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>2..3</td><td>31:0</td><td><b>Immediate Data</b><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

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**
	21	**Store Data Index** Format: | U1 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 hardware status page.
	20:19	**Reserved** Format: | MBZ
	18	**TLB Invalidate** Format: | U1 **Description** 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. If GFX_MODE(0x229c) bit 13, this command will cause a config write to MMIO register space with the address 0x4f100.
	17	**Synchronize GFDT surface** Format: | U1 If enabled, at the end of the current flush the last level cache is cleared of all the cachelines which have been marked with the special GFDT flags. Store DW must be enabled
	16	**Reserved** Format: | MBZ
	15:14	**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.

<table>
<tr><th colspan="3">MI_FLUSH_DW</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td>1h</td><td>Write Immediate Data</td><td>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</td></tr>
<tr><td>2h</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>3h</td><td></td><td>Write the TIMESTAMP register to the Destination Address with a granularity of 80ns.
The upper 28 bits of the TIMESTAMP register are tied to '0'.</td></tr>
</table>
</td></tr>
<tr><td></td><td>13:10</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table>
</td></tr>
<tr><td></td><td>9</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table>
</td></tr>
<tr><td></td><td>8</td><td><b>Notify Enable</b>
<table><tr><td>Format:</td><td>U1</td></tr></table>
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.
</td></tr>
<tr><td></td><td>7</td><td><b>Video Pipeline Cache invalidate</b>
<table><tr><td>Format:</td><td>U1</td></tr></table>
Enable the invalidation of the video cache at the end of this flush
</td></tr>
<tr><td></td><td>6</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>5:0</td><td><b>DWord Length</b>
<table><tr><td>Format:</td><td>=n Total Length - 2</td></tr></table>
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>2h</td><td>Excludes DWord (0,1) = 1 for DWord, 2 for QWord <b>[Default]</b></td></tr>
</table>
</td></tr>
<tr><td>1</td><td>31:3</td><td><b>Address</b>
<table><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><b>Destination Address Type</b>
Defines address space of Destination Address
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>PPGTT</td><td>Use PPGTT address space for DW write</td></tr>
<tr><td>1h</td><td>GGTT</td><td>Use GGTT address space for DW write</td></tr>
</table>
</td></tr>
</table>

MI_FLUSH_DW

		Programming Notes Ignored if "No write" is the selected in Operation.
	1:0	**Reserved** Format: | MBZ
2..3	31:0	**Immediate Data** 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

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.

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** 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).
	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

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).

Programming Notes
A stalling flush must be sent down pipeline before issuing this command. 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. 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.
MI_LOAD_REGISTER_IMM command to program Scanline Register followed by Wait For Event command with Scanline Wait, should always be programmed in the same cacheline together without any commands (including pipe control) in between and also should be submitted in the same ring dispatch.

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**
	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).

MI_LOAD_REGISTER_IMM		
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

Source:	VideoCS
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.

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** 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: 0h Excludes DWord (0,1) Format: =n Total Length - 2
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). Mapped
	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

Source:	RenderCS, BlitterCS, VideoCS
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 LRIs: • 0x8800 - 0x88FF • >= 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.
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 "16" dummy MI_STORE_DATA_IMM (write to scratch space) commands prior to programming of this command. Example: MI_STORE_REGISTE_MEM (0x2288, 0x2CF0_0000) .... MI_STORE_DATA_IMM (16 times) (Dummy data, Scratch Address) MI_LOAD_REGISTER_MEM(0x2288, 0x2CF0_0000)

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="5">0</td><td>31:29</td><td><b>Command Type</b>
<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><b>MI Command Opcode</b>
<table>
<tr><td>Default Value:</td><td>29h MI_LOAD_REGISTER_MEM</td></tr>
<tr><td>Format:</td><td>OpCode</td></tr>
</table></td></tr>
<tr><td>22</td><td><b>Use Global GTT</b>
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>This command will use the per process GTT to translate the Address.</td></tr>
<tr><td>1h</td><td>Global Graphics Address</td><td>This command will use the global GTT to translate the Address.</td></tr>
</table></td></tr>
<tr><td>21</td><td><b>Async Mode Enable</b>
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.</td></tr>
<tr><td>20:8</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
</table>

MI_LOAD_REGISTER_MEM

	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 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.
	1:0	**Reserved** Format: MBZ
2	31:2	**Memory Address** Format: GraphicsAddress[31:2] 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[31:2] for a DWord register
	1:0	**Reserved** Format: MBZ

MI_NOOP

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).

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>0h MI_COMMAND</td></tr></table></td></tr>
<tr><td></td><td>28:23</td><td><b>MI Command Opcode</b><table><tr><td>Default Value:</td><td>0h MI_NOOP</td></tr></table></td></tr>
<tr><td></td><td>22</td><td><b>Identification Number Register Write Enable</b><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></td><td>21:0</td><td><b>Identification Number</b><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

Source:	RenderCS
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).

Performance
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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b>
<table><tr><td>Default Value:</td><td>0h MI_COMMAND</td></tr></table></td></tr>
<tr><td></td><td>28:23</td><td><b>MI Command Opcode</b>
<table><tr><td>Default Value:</td><td>0h MI_NOOP</td></tr></table></td></tr>
<tr><td></td><td>22</td><td><b>Identification Number Register Write Enable</b>
<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></td><td>21:0</td><td><b>Identification Number</b>
<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

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

Source: RenderCS
Length Bias: 1

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b>
<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><b>MI Command Opcode</b>
<table>
<tr><td>Default Value:</td><td>0Ch MI_PREDICATE</td></tr>
<tr><td>Format:</td><td>OpCode</td></tr>
</table></td></tr>
<tr><td></td><td>22:8</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>7:6</td><td><b>Load Operation</b>
This field controls if/how the Predicate state bit is modified.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>LOADOP_KEEP</td><td>The Predicate state bit is unmodified.</td></tr>
<tr><td>1h</td><td>Reserved</td><td></td></tr>
<tr><td>2h</td><td>LOADOP_LOAD</td><td>The Predicate state bit is loaded with the combine operation result.</td></tr>
<tr><td>3h</td><td>LOADOP_LOADINV</td><td>The Predicate state bit is loaded with the inverted combine operation result.</td></tr>
</table></td></tr>
<tr><td></td><td>5</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>4:3</td><td><b>Combine Operation</b>
This field controls if/how the result of the compare operation is combined with the current Predicate state bit.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>COMBINEOP_SET</td><td>The combine operation output the compare result unmodified.</td></tr>
<tr><td>1h</td><td>COMBINEOP_AND</td><td>The combine operation outputs the AND of the compare result and the current Predicate state bit.</td></tr>
<tr><td>2h</td><td>COMBINEOP_OR</td><td>The combine operation outputs the OR of the compare result and the current Predicate state bit.</td></tr>
<tr><td>3h</td><td>COMBINEOP_XOR</td><td>The combine operation outputs the XOR of the compare result and the current Predicate state bit.</td></tr>
</table></td></tr>
<tr><td></td><td>2</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>1:0</td><td><b>Compare Operation</b>
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.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
</table></td></tr>
</table>

MI_PREDICATE

		0h	COMPAREOP_TRUE	The compare operation outputs TRUE. The PredicateData register is unmodified.
		1h	COMPAREOP_FALSE	The compare operation outputs FALSE. The PredicateData register is unmodified.
		2h	COMPAREOP_SRCS_EQUAL	(MItemp0 - MItemp1) is computed and loaded into the PredicateData register. The compare operation outputs (MItemp0 == MItemp1).
		3h	COMPAREOP_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

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.

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
	28:23	**MI Command Opcode** Default Value: 07h MI_REPORT_HEAD
	22:0	**Reserved** Format: MBZ

MI_REPORT_HEAD

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. 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

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.

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_SEMAPHORE_MBOX

Source: BlitterCS

Length Bias: 2

Description
MI_SEMAPHORE_MBOX command provides capability in Blitter Engine to wait conditionally until a given synchronization register gets updated with a value greater than the "SEMAPHORE_DATA_DWORD" mentioned inline in this command. Synchronization registers can be updated through CPU MMIO access or through execution of MI_LOAD_REGISTER_IMM command in other engines. Synchronization between contexts (especially between contexts running on 2 different engines) is provided by the MI_SEMAPHORE_MBOX command.
If execution is stalled due to this command, the engine will specify that the engine is IDLE to the power management engine.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b>
<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><b>MI Command Opcode</b>
<table>
<tr><td>Default Value:</td><td>16h MI_SEMAPHORE_MBOX</td></tr>
<tr><td>Format:</td><td>OpCode</td></tr>
</table></td></tr>
<tr><td></td><td>22:21</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>20</td><td><b>Reserved</b>
<table>
<tr><td>Default Value:</td><td>1h</td></tr>
<tr><td>Format:</td><td>Must Be One</td></tr>
</table></td></tr>
<tr><td></td><td>19</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>18</td><td><b>Reserved</b>
<table>
<tr><td>Default Value:</td><td>1h</td></tr>
<tr><td>Format:</td><td>Must Be One</td></tr>
</table></td></tr>
<tr><td></td><td>17:16</td><td><b>Register Select</b>
This field indicates the synchronization register to be used for comparison with the inline data.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0h</td><td>CS register (BRSYNC)</td></tr>
<tr><td>1h</td><td>Reserved</td></tr>
<tr><td>2h</td><td>VCS regiser (BVSYNC)</td></tr>
<tr><td>3h</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td></td><td>15:8</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b></td></tr>
</table>

MI_SEMAPHORE_MBOX

		Default Value: 1h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:0	**Semaphore Data Dword** Format: U32 Inline Data Dword to compare with the selected synchronization register. The Data dword is supplied by software to control execution of the command buffer. If the data in the selected synchronization register is greater than this dword, the execution of the command buffer continues.
2	31:0	**Reserved** Format: MBZ

MI_SEMAPHORE_MBOX

Source:	RenderCS
Length Bias:	2

MI_SEMAPHORE_MBOX command provides capability in Render Engine to wait conditionally until a given synchronization register gets updated with a value greater than the "SEMAPHORE_DATA_DWORD" mentioned inline in this command. Synchronization registers can be updated through CPU MMIO access or through execution of MI_LOAD_REGISTER_IMM command in other engines.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><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><b>MI Command Opcode</b><table><tr><td>Default Value:</td><td>16h MI_SEMAPHORE_MBOX</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>22:21</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>20</td><td><b>Reserved</b><table><tr><td>Default Value:</td><td>1h</td></tr><tr><td>Format:</td><td>Must Be One</td></tr></table></td></tr>
<tr><td></td><td>19</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>18</td><td><b>Reserved</b><table><tr><td>Default Value:</td><td>1h</td></tr><tr><td>Format:</td><td>Must Be One</td></tr></table></td></tr>
<tr><td></td><td>17:16</td><td><b>Register Select</b>
This field indicates the synchronization register to be used for comparison with the inline data.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>RVSYNC</td><td>VCS Register</td></tr><tr><td>1h</td><td>Reserved</td><td>Reserved</td></tr><tr><td>2h</td><td>RBSYNC</td><td>BCS Register</td></tr><tr><td>3h</td><td>Reserved</td><td>Reserved</td></tr></table></td></tr>
<tr><td></td><td>15:14</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>13:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>1h</td></tr><tr><td>Format:</td><td>=n Total Length - 2. Excludes DWord (0,1).</td></tr></table></td></tr>
<tr><td>1</td><td>31:0</td><td><b>Semaphore Data Dword</b></td></tr>
</table>

MI_SEMAPHORE_MBOX

		Format: U32 Inline Data Dword to compare with the selected synchronization register. The Data dword is supplied by software to control execution of the command buffer. If the data in the selected synchronization register is greater than this dword, the execution of the command buffer continues.
2	31:0	**Reserved** Format: MBZ

MI_SEMAPHORE_MBOX

Source:	VideoCS
Length Bias:	2

Description
MI_SEMAPHORE_MBOX command provides capability in Video Engine to wait conditionally until a given synchronization register gets updated with a value greater than the "SEMAPHORE_DATA_DWORD" mentioned inline in this command. Synchronization registers can be updated through CPU MMIO access or through execution of MI_LOAD_REGISTER_IMM command in other engines.
If execution is stalled due to this command, the engine will specify that the engine is IDLE to the power management engine.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><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><b>MI Command Opcode</b><table><tr><td>Default Value:</td><td>16h MI_SEMAPHORE_MBOX</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>22:21</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>20</td><td><b>Reserved</b><table><tr><td>Default Value:</td><td>1h</td></tr><tr><td>Format:</td><td>Must Be One</td></tr></table></td></tr>
<tr><td></td><td>19</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>18</td><td><b>Reserved</b><table><tr><td>Default Value:</td><td>1h</td></tr><tr><td>Format:</td><td>Must Be One</td></tr></table></td></tr>
<tr><td></td><td>17:16</td><td><b>Register Select</b>
This field indicates the synchronization register to be used for comparison with the inline data.<table><tr><th>Value</th><th>Name</th></tr><tr><td>0h</td><td>BCS register (VBSYNC)</td></tr><tr><td>1h</td><td>Reserved</td></tr><tr><td>2h</td><td>CS register (VRSYNC)</td></tr><tr><td>3h</td><td>Reserved</td></tr></table></td></tr>
<tr><td></td><td>15:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>1h Excludes DWord (0,1)</td></tr></table></td></tr>
</table>

MI_SEMAPHORE_MBOX

		Format: =n Total Length - 2
1	31:0	**Semaphore Data Dword** Format: U32 Inline Data Dword to compare with the selected synchronization register. The Data dword is supplied by software to control execution of the command buffer. If the data in the selected synchronization register is greater than this dword, the execution of the command buffer continues.
2	31:0	**Reserved** Format: MBZ

MI_SET_CONTEXT

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.
- This command needs to be always followed by a single MI_NOOP instruction to workaround a silicon issue.
- 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

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** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Format: =n Total Length - 2. Excludes DWord (0,1).

MI_SET_CONTEXT

1	31:12	**Logical Context Address** 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** Format: MBZ
	8	**Reserved, Must be 1** Format: Must Be One
	7:5	**Reserved** Format: MBZ
	4	**Reserved** Format: MBZ
	3	**Extended State Save Enable** Format: Enable If set, the extended 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). This bit must be 1 when RS2 power state is enabled (via MCHBAR, offset 0x11B8)
	2	**Extended State Restore Enable** Format: Enable If set, the extended 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 method can be used to restore things such as filter coefficients using the indirect state restore followed by a restore of the extended logical context data. 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). This bit must be 1 when RS2 power state is enabled (via MCHBAR, offset 0x11B8)
	1	**Force Restore** When switching to this logical context a comparison between Logical Context Address and the

MI_SET_CONTEXT

		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_STORE_DATA_IMM

Source:	RenderCS
Length Bias:	2

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).

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.

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: Boolean If set, this command will use the global GTT to translate the Address. If clear, the PPGTT will be used. This bit must be '1' if the Per Process GTT Enable bit is clear.
	21	**Reserved** Format: MBZ
	20:10	**Reserved** Format: MBZ
	9:6	**Reserved** Format: MBZ
	5:0	**DWord Length** Default Value: 2h Excludes DWord (0,1) Format: =n Total Length - 2. Excludes DWord (0,1) **Programming Notes** Dword Length programmed must not exceed 0x3.
1	31:0	**Reserved** Format: MBZ
2	31:2	**Address**

MI_STORE_DATA_IMM

		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
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

Source:	BlitterCS
Length Bias:	2

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).

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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>0h MI_COMMAND</td></tr></table></td></tr>
<tr><td></td><td>28:23</td><td><b>MI Command Opcode</b><table><tr><td>Default Value:</td><td>20h MI_STORE_DATA_IMM</td></tr></table></td></tr>
<tr><td></td><td>22</td><td><b>Use Global GTT</b>
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>This command will use the per process GTT to translate the Address</td></tr><tr><td>1h</td><td>Global Graphics Address</td><td>This command will use the global GTT to translate the Address.</td></tr></table></td></tr>
<tr><td></td><td>21</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>20:10</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>9:0</td><td><b>DWord Length</b><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>1</td><td>31:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>2</td><td>31:2</td><td><b>Address</b><table><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>
</table>

MI_STORE_DATA_IMM

	1:0	**Reserved** Format: MBZ
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

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. 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.

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. If clear, the PPGTT will be used. This bit must be '1' if the Per Process GTT Enable bit is clear.
	21:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) = 3 for QWord, 2 for DWord Format: =n Total Length - 2
1	31:0	**Reserved** Format: MBZ
2	31:2	**Address** Format: GraphicsAddress[31:2] 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
3	31:0	**Data DWord 0**

MI_STORE_DATA_IMM

		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_INDEX

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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>0h MI_COMMAND</td></tr></table></td></tr>
<tr><td></td><td>28:23</td><td><b>MI Command Opcode</b><table><tr><td>Default Value:</td><td>21h MI_STORE_DATA_INDEX</td></tr></table></td></tr>
<tr><td></td><td>22</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>21</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>20:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>1h Excludes DWord (0,1) = 1 for DWord, 2 for QWord</td></tr><tr><td>Format:</td><td>=n Total Length - 2</td></tr></table></td></tr>
<tr><td>1</td><td>31:12</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11:2</td><td><b>Offset</b><table><tr><td>Format:</td><td>U10 zero-based DWord offset into the HW status page.</td></tr><tr><td>Format:</td><td>HardwareStatusPageOffset[11:2]U32</td></tr></table>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.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[16, 1023]</td><td></td></tr></table></td></tr>
<tr><td></td><td>1:0</td><td><b>Reserved</b></td></tr>
</table>

MI_STORE_DATA_INDEX

		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

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**
	21	**Reserved** Format: MBZ
	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.
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.

<table>
<tr><th colspan="3">MI_STORE_DATA_INDEX</th></tr>
<tr><td></td><td></td><td><table><tr><th>Value</th><th>Name</th></tr><tr><td>[16, 1023]</td><td></td></tr></table></td></tr>
<tr><td></td><td>1:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>2</td><td>31:0</td><td><b>Data DWord 0</b><table><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>
<tr><td>3</td><td>31:0</td><td><b>Data DWord 1</b><table><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_INDEX

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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><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><b>MI Command Opcode</b><table><tr><td>Default Value:</td><td>21h MI_STORE_DATA_INDEX</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>22</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>21</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>20:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>0h Excludes DWord (0,1) = 2 for QWord</td></tr><tr><td>Format:</td><td>=n Total Length - 2</td></tr></table></td></tr>
<tr><td>1</td><td>31:12</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11:2</td><td><b>Offset</b><table><tr><td>Format:</td><td>U10 zero-based DWord offset into the HW status page</td></tr><tr><td>Format:</td><td>GraphicsAddress[11:2]U32</td></tr></table>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></table></td></tr>
</table>

MI_STORE_DATA_INDEX

		[16, 1023] **Programming Notes** The first few DWords of this status page are reserved for special-purpose data storage - targeting these reserved locations via this command is UNDEFINED.
	1:0	**Reserved** Format: MBZ
2	31:0	**Data DWord 0** Format: U32 FormatDesc 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 FormatDesc This field specifies the upper DWord value to be written to the targeted QWord location (DW 1).

MI_STORE_REGISTER_MEM

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

- The command temporarily halts command execution.
- The memory address for the write is snooped on the host bus.
- This command will cause undefined data to be written to memory if given register addresses for the PGTBL_CTL_0 or FENCE registers.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b>
<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><b>MI Command Opcode</b>
<table><tr><td>Default Value:</td><td>24h MI_STORE_REGISTER_MEM</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>22</td><td><b>Use Global GTT</b>
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>This command will use the per process GTT to translate the Address.</td></tr><tr><td>1h</td><td>Global Graphics Address</td><td>This command will use the global GTT to translate the Address.</td></tr></table></td></tr>
<tr><td></td><td>21</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>20:8</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b>
<table><tr><td>Default Value:</td><td>1h Excludes DWord (0,1)</td></tr><tr><td>Format:</td><td>=n Total Length - 2</td></tr></table></td></tr>
<tr><td>1</td><td>31:23</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>22:2</td><td><b>Register Address</b>
<table><tr><td>Format:</td><td>MMIOAddress[22:2]MMIO_Register</td></tr></table>
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.</td></tr>
</table>

MI_STORE_REGISTER_MEM

		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.
	1:0	**Reserved** Format: MBZ
2	31:2	**Memory Address** Format: GraphicsAddress[31:2]MMIO_Register 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[31:2] for a DWord register
	1:0	**Reserved** Format: MBZ

MI_TOPOLOGY_FILTER

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_PrimTopoType 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_URB_CLEAR

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.
MI_URB_CLEAR must be programmed following MI_SET_CONTEXT and before workload is submitted, when a given context expects URB locations to be initialized to 0x0.

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**	
		Format:	MBZ
	29	**Reserved**	
		Format:	MBZ
	28:16	**URB Clear Length**	
		This field specifies the number of 256b entries in the URB to be cleared to zero.	
		Value	**Name**
		[0,8191]	
	15	**Reserved**	
		Format:	MBZ
	14	**Reserved**	
		Format:	MBZ
	13:0	**URB Address**	
		Format:	URBAddress[18:5] 256b aligned
		This field specifies Bits 18:5 of the URB Address	

MI_USER_INTERRUPT

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** Format: MBZ

MI_USER_INTERRUPT

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

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

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).

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b>
<table><tr><td>Default Value:</td><td>0h MI_COMMAND</td></tr></table></td></tr>
<tr><td></td><td>28:23</td><td><b>MI Command Opcode</b>
<table><tr><td>Default Value:</td><td>03h MI_WAIT_FOR_EVENT</td></tr></table></td></tr>
<tr><td></td><td>22</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>21</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>20</td><td><b>Display Sprite C Flip Pending Wait Enable</b>
<table><tr><td>Format:</td><td>Enable</td></tr></table>
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).</td></tr>
<tr><td></td><td>19:16</td><td><b>Condition Code Wait Select</b>
This field enables a wait for the duration that the corresponding condition code is active. These enable select one of 15 condition codes in the EXCC register, that cause the parser to wait until that condition-code in the EXCC is cleared.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Not Enabled</td><td>Condition Code Wait not enabled</td></tr>
<tr><td>1h-5h</td><td>Enabled</td><td>Condition Code select enabled; selects one of 5 codes, 0 - 4</td></tr>
<tr><td>6h-15h</td><td>Reserved</td><td></td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>Note that not all condition codes are implemented. The parser operation is UNDEFINED if an unimplemented condition code is selected by this field. The description of the EXCC register (Memory Interface Registers) lists the codes that are implemented.</td></tr>
</table></td></tr>
<tr><td></td><td>15</td><td><b>Display Plane C Flip Pending Wait Enable</b>
<table><tr><td>Format:</td><td>Enable</td></tr></table>
This field enables a wait for the duration of a Display Plane C "Flip Pending" condition. If a flip</td></tr>
</table>

MI_WAIT_FOR_EVENT

		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).
	14	**Reserved** Format: MBZ
	13:12	**Reserved** Format: MBZ
	11	**Reserved** Format: MBZ
	10	**Display Sprite B Flip Pending Wait Enable** 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** 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).
	8	**Reserved** Format: MBZ
	7:6	**Reserved** Format: MBZ
	5:4	**Reserved** Format: MBZ
	3	**Reserved** Format: MBZ
	2	**Display Sprite A Flip Pending Wait Enable** 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).
	1	**Display Plane A Flip Pending Wait Enable** 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).

MI_WAIT_FOR_EVENT		
	0	**Reserved** Format: MBZ

MI_WAIT_FOR_EVENT

Source:	VideoCS
Length Bias:	1

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. 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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b>
<table><tr><td>Default Value:</td><td>0h MI_COMMAND</td></tr></table></td></tr>
<tr><td></td><td>28:23</td><td><b>MI Command Opcode</b>
<table><tr><td>Default Value:</td><td>03h MI_WAIT_FOR_EVENT</td></tr></table></td></tr>
<tr><td></td><td>22:20</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>19:16</td><td><b>Condition Code Wait Select</b>
<table><tr><td>This field enables a wait for the duration that the corresponding condition code is active. These enable select one of 15 condition codes in the EXCC register, that cause the parser to wait until that condition-code in the EXCC is cleared.</td></tr><tr><td></td></tr></table>
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Not enabled</td><td>Condition Code Wait Not Enabled</td></tr>
<tr><td>1h-5h</td><td>Enable</td><td>Condition Code select enabled; selects one of 5 codes, 0 - 4</td></tr>
<tr><td>6h-15h</td><td>Reserved</td><td></td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>Note that not all condition codes are implemented. The parser operation is UNDEFINED if an unimplemented condition code is selected by this field. The description of the EXCC register (Memory Interface Registers) lists the codes that are implemented.</td></tr>
</table></td></tr>
<tr><td></td><td>15:0</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
</table>

MI_WAIT_FOR_EVENT

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.

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	**Display Pipe C Horizontal Blank Wait Enable** Format: Enable This field enables a wait until the start of next Display Pipe C Horizontal Blank event occurs. This event is described as the start of the next Display C Horizontal blank period. Note that this can cause a wait for up to a line.
	21	**Display Pipe C Vertical Blank Wait Enable** Format: Enable

MI_WAIT_FOR_EVENT

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.

20 **Display Sprite C Flip Pending Wait Enable**

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).

19:16 **Condition Code Wait Select**

This field enables a wait for the duration that the corresponding condition code is active. These enable select one of 15 condition codes in the EXCC register, that cause the parser to wait until that condition-code in the EXCC is cleared.

Value	Name	Description
0h	Not enabled	Condition Code Wait Not Enabled
1h-5h	Enable	Condition Code Select Enabled; selects one of 5 codes, 0 - 4
6h-15h	Reserved	

Programming Notes
Note that not all condition codes are implemented. The parser operation is UNDEFINED if an unimplemented condition code is selected by this field. The description of the EXCC register (Memory Interface Registers) lists the codes that are implemented.

15 **Display Plane C Flip Pending Wait Enable**

Format:	Enable

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).

14 **Display Pipe C Scan Line Wait Enable**

Format:	Enable

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.

13 **Display Pipe B Horizontal Blank Wait Enable**

Format:	Enable

This field enables a wait until the start of next Display Pipe B "Horizontal Blank" event occurs. This event is described as the start of the next Display B Horizontal blank period. Note that this can cause a wait for up to a line.

12 **Reserved**

Format:	MBZ

MI_WAIT_FOR_EVENT

	11	**Display Pipe B Vertical Blank Wait Enable** 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** 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** 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.
	8	**Display Pipe B Scan Line Wait Enable** Format: | Enable 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.
	7:6	**Reserved** Format: | MBZ
	5	**Display Pipe A Horizontal Blank Wait Enable** Format: | Enable This field enables a wait until the start of next Display Pipe A Horizontal Blank event occurs. This event is described as the start of the next Display A Horizontal blank period. Note that this can cause a wait for up to a line.
	4	**Reserved** Format: | MBZ
	3	**Display Pipe A Vertical Blank Wait Enable** 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** Format: | Enable This field enables a wait for the duration of a Display Sprite A "Flip Pending" condition. If a flip

<table>
<tr><th colspan="3">MI_WAIT_FOR_EVENT</th></tr>
<tr><td></td><td></td><td>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).</td></tr>
<tr><td></td><td>1</td><td><b>Display Plane A Flip Pending Wait Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>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).</td></tr>
<tr><td></td><td>0</td><td><b>Display Pipe A Scan Line Wait Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>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.</td></tr>
</table>

mov - Move

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.

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.

Restriction
Restriction: Raw move is not supported for Float values in ALT mode if any values are infinities or NaNs.
Restriction: An accumulator can be a source or destination operand but not both.
Restriction: If the source type and destination type differ, conditional modifiers are not allowed.

Syntax

```
[(pred)] mov[.cmod] (exec_size) reg reg [(pred)] mov[.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	Y	Y	Y

Src Types	Dst Types
*B,*W,*D	*B,*W,*D
*B,*W,*D	F
F	*B,*W,*D
F	F
*W,*D	DF
F	DF
DF	*W,*D
DF	F

mov - Move

DF	DF

DWord	Bit	Description
0..3	127:64	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32
	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

movi - Move Indexed

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
Restriction: Source operand cannot be accumulators. The source operand must be a general register.
Restriction: The source and destination must have the same type.
Restriction: The execution size must be <= 8 (1, 2, 4, or 8).
Restriction: The address register for the source must be aligned to the base (a0.0).
Restriction: The destination register (directly or indirectly addressed) must be 16-byte aligned.
Restriction: The destination region (directly or indirectly addressed) must point to the same GRF register.
Restriction: The destination stride in bytes must equal the source element size in bytes.
Restriction: The Align16 access mode is not allowed.
Restriction: All the index registers (address subregisters) used must point to the same GRF register.
Restriction: The instruction must use 1x1 indirect regioning.
Restriction: The destination offset is only used to create channel enables. Each element of the

movi - Move Indexed

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.

Restriction: 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	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32
	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

mul - Multiply

Source: EuIsa

Length Bias: 4

Description

The mul instruction performs component-wise multiplication of src0 and src1 and stores the results in dst.

When both src0 and src1 are of type D or UD, only the low 16 bits of each element of src1 are used. The macro described in the mach instruction should be used to obtain the full precision 64-bit multiplication result.

Multiplication of two floating-point numbers follows the rules in mul - Multiply) based on the applicable floating-point mode.

Format:
[(pred)] mul[.cmod] (exec_size) dst src0 src1

Restriction

Restriction: Use a source modifier with add to implement subtraction.

Restriction: When operating on integers with at least one of the source being a DWord type (signed or unsigned), the destination cannot be floating-point (implementation note: the data converter only looks at the low 34 bits of the result).

Restriction: When operating on integers with at least one source having a DWord type (signed or unsigned), the Overflow and Sign flags are undefined. Therefore, conditional modifiers and saturation (.sat) cannot be used in this case.

Restriction: When multiplying a DW and a W, the W has to be on src1, and the DW has to be on src0.

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
B	B
B	W
B	D
W	W
W	D
W,D	D
F	F

mul - Multiply

DF	DF

DWord	Bit	Description
0..3	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM
	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

mac - Multiply Accumulate

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

Restriction

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
*B,*W	*B,*W,*D
F	F
DF	DF

DWord	Bit	Description
0..3	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM
	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

mach - Multiply Accumulate High

Source:	EuIsa
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 src1 with the high word of the DWord in src0, 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 four 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<8;8,1>:d //All channels must be enabled

mach (8) rTemp<1>:d r2.0<8;8,1>:d r3.0<8;8,1>:d //All channels must be enabled

mov (8) r5.0<1>:d rTemp<8;8,1>:d // High 32 bits
mov (8) r6.0<1>:d acc0:d // Low 32 bits
```

The mul and mach instructions must have all channels enabled. The first mov should have channel enable from the destHI of IMUL, the second 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
Restriction: Accumulator is an implicit source and thus cannot be an explicit source operand.
Restriction: AccWrEn is required. The accumulator is an implicit destination and thus cannot be an explicit destination operand.

Syntax
`[(pred)] mach[.cmod] (exec_size) reg reg reg [(pred)] mach[.cmod] (exec_size) reg reg imm32`

mach - Multiply Accumulate High

Pseudocode

Evaluate(WrEn);

```
for ( n = 0; n < exec_size; n++ ) {
    acc.chan[n][63:0] = (src0.chan[n][31:16] * src1.chan[n][31:0]) << 16 + acc.chan[n][63:0];
    if ( WrEn.chan[n] ) {
        dst.chan[n][31:0] = acc.chan[n][63:32];
    }
}
```

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	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM
	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

mad - Multiply Add

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
Restriction: No explicit accumulator access because this is a three-source instruction. AccWrEn is allowed for implicitly updating the accumulator.
Restriction: All three-source instructions have certain restrictions, described in Instruction Machine Formats.

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
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
	105	**Reserved** Format: MBZ
	104:85	**Source 1** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC
	84	**Reserved** Format: MBZ
	83:64	**Source 0** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC
	63:56	**Destination Register Number** Format: DstRegNum

mad - Multiply Add

<table>
<tr><td></td><td>55:53</td><td><b>Destination Subregister Number</b><table><tr><td>Format:</td><td>DstSubRegNum[2:0]</td></tr></table></td></tr>
<tr><td></td><td>52:49</td><td><b>Destination Channel Enable</b><table><tr><td>Format:</td><td>ChanEn[4]</td></tr></table>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 <i>x</i>, <i>y</i>, <i>z</i>, and <i>w</i>, respectively, where <i>x</i> corresponds to Channel 0 in the group and <i>w</i> corresponds to channel 3 in the group</td></tr>
<tr><td></td><td>48</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>47</td><td><b>NibCtrl</b><table><tr><td>Format:</td><td>NibCtrl</td></tr></table></td></tr>
<tr><td></td><td>46</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>45:44</td><td><b>Destination Data Type</b>
This field contains the data type for the destination<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>Single Precision Float</td></tr><tr><td>01b</td><td>DWord</td></tr><tr><td>10b</td><td>Unsigned DWord</td></tr><tr><td>11b</td><td>Double Precision Float</td></tr></table></td></tr>
<tr><td></td><td>43:42</td><td><b>Source Data Type</b>
This field contains the data type for all three sources<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>Single Precision Float</td></tr><tr><td>01b</td><td>DWord</td></tr><tr><td>10b</td><td>Unsigned DWord</td></tr><tr><td>11b</td><td>Double Precision Float</td></tr></table></td></tr>
<tr><td></td><td>41:40</td><td><b>Source 2 Modifier</b><table><tr><td>Exists If:</td><td>([Property[Source Modification]=='true')</td></tr><tr><td>Format:</td><td>SrcMod</td></tr></table></td></tr>
<tr><td></td><td>39:38</td><td><b>Source 1 Modifier</b><table><tr><td>Exists If:</td><td>([Property[Source Modification]=='true')</td></tr><tr><td>Format:</td><td>SrcMod</td></tr></table></td></tr>
<tr><td></td><td>41:36</td><td><b>Reserved</b><table><tr><td>Exists If:</td><td>([Property[Source Modification]=='false')</td></tr></table></td></tr>
</table>

mad - Multiply Add

		Format:	MBZ
	37:36	**Source 0 Modifier**	
		Exists If:	([Property[Source Modification]=='true')
		Format:	SrcMod
	35	**Reserved**	
		Format:	MBZ
	34	**Flag Register Number** This field contains the flag register number for instructions with a non-zero Conditional Modifier.	
	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

nop - No Operation

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
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	**DebugCtrl** This field allows the insertion of a breakpoint at the current instruction. When the bit is set, hardware automatically stores the current IP in CR register and jumps to the System IP (SIP) BEFORE executing the current instruction. Value 0: No Breakpoint **[Default]** Value 1: Breakpoint
	29:7	**Reserved** Format: MBZ
	6:0	**Opcode** Format: EU_OPCODE

PIPE_CONTROL

Source: RenderCS
Length Bias: 2

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

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h GFXPIPE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Command SubType</b><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></td><td>26:24</td><td><b>3D Command Opcode</b><table><tr><td>Default Value:</td><td>2h PIPE_CONTROL</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>23:16</td><td><b>3D Command Sub Opcode</b><table><tr><td>Default Value:</td><td>0h PIPE_CONTROL</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>15:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>3h DWORD_COUNT_n</td></tr><tr><td>Format:</td><td>=n</td></tr></table></td></tr>
<tr><td>1</td><td>31:28</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>27</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>26</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>25</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>24</td><td><b>Destination Address Type</b>
Defines address space of Destination Address<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>PPGTT</td><td>Use PPGTT address space for DW write</td></tr><tr><td>1h</td><td>GGTT</td><td>Use GGTT address space for DW write</td></tr></table><table><tr><th>Programming Notes</th></tr><tr><td>Ignored if ""No Write" is selected in Operation.</td></tr></table></td></tr>
</table>

PIPE_CONTROL

23 **LRI Post Sync Operation**

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

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

22 **Reserved**

21 **Store Data Index**

Format:	U1

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).

20 **Command Streamer Stall Enable**

Format:	U1

If ENABLED, the sync operation will not occur until all previous flush operations pending a completion of those previous flushes will complete, including the flush produced from this command. This enables the command to act similar to the legacy MI_FLUSH command.

Programming Notes
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)

19 **Reserved**

18 **TLB Invalidate**

Format:	U1

If ENABLED, all TLBs belonging to Render Engine will be invalidated once the flush operation is complete. Note that if the flush TLB invalidation mode is clear, a TLB invalidate will occur irrespective of this bit setting
If ENABLED, PIPE_CONTROL command will flush the in flight data written out by render engine to Global Observation point on flush done. Also Requires stall bit ([20] of DW1) set.

PIPE_CONTROL

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**

Format:	MBZ

16 **Generic Media State Clear**

Format:	Disable

If set, all generic media state context information will not be included with the next context save, assuming no new state is initiated after the flush. If clear, the generic media state context save state will not be affected. An MI_FLUSH with this bit set should be issued once all the Media Objects that will be processed by a given persistent root thread have been issued or when an MI_SET_CONTEXT switching from a generic media context to a 3D context completes. When using MI_SET_CONTEXT, once state is programmed, it will be saved and restarted as part of any context each time that context is saved/restored until an MI_FLUSH with this bit set is issued in that context.

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
0h	No Write	No write occurs as a result of this instruction. This can be used to implement a "trap" operation, etc.
1h	Write Immediate Data	Write the QWord containing Immediate Data Low, High DWs to the Destination Address
2h	Write PS Depth Count	Write the 64-bit PS_DEPTH_COUNT register to the Destination Address
3h	Write Timestamp	Write the 64-bit TIMESTAMP register to the Destination Address

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

13 **Depth Stall Enable**

Format:	Enable

This bit should 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

PIPE_CONTROL

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

12 **Render Target Cache Flush Enable**

Format:	Enable

Setting this bit will force Render Cache to be flushed to memory prior to this synchronization point completing. This bit should 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
0h	Disable Flush	Render Target Cache is NOT flushed.
1h	Enable Flush	Render Target Cache is flushed.

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

11 **Instruction Cache Invalidate Enable**

Format:	Enable

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

10 **Texture Cache Invalidation Enable**

Format:	Enable

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

9 **Indirect State Pointers Disable**

Format:	Enable

Description
At the completion of the post-sync operation associated with this pipe control packet, the indirect state pointers in the hardware are considered invalid; the indirect pointers are not saved in the context. If any new indirect state commands are executed in the command stream while the pipe control is pending, the new indirect state commands are preserved.

<table>
<tr><th colspan="3">PIPE_CONTROL</th></tr>
<tr><td></td><td></td><td><table><tr><td>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.</td></tr></table></td></tr>
<tr><td></td><td>8</td><td><b>Notify Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>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.</td></tr>
<tr><td></td><td>7</td><td><b>Pipe Control Flush Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>If ENABLED, the PIPE_CONTROL command will wait until all previous writes of immediate data from post sync circles are complete before executing the next command.</td></tr>
<tr><td></td><td>6</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>5</td><td><b>DC Flush Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>Setting this bit enables flushing of the L3$ portions that caches DC writes.</td></tr>
<tr><td></td><td>4</td><td><b>VF Cache Invalidation Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>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.</td></tr>
<tr><td></td><td>3</td><td><b>Constant Cache Invalidation Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>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.</td></tr>
<tr><td></td><td>2</td><td><b>State Cache Invalidation Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>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.</td></tr>
<tr><td></td><td>1</td><td><b>Stall At Pixel Scoreboard</b><table><tr><td>Format:</td><td>Enable</td></tr></table>Defines the behavior of PIPE_CONTROL command at the pixel scoreboard.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Disable</td><td>Stall at the pixel scoreboard is disabled.</td></tr><tr><td>1h</td><td>Enable</td><td>Stall at the pixel scoreboard is enabled.</td></tr></table></td></tr>
</table>

<table>
<tr><th colspan="3">PIPE_CONTROL</th></tr>
<tr><td></td><td></td><td><table><tr><th>Programming Notes</th></tr><tr><td>This bit should 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.</td></tr></table></td></tr>
<tr><td></td><td>0</td><td><b>Depth Cache Flush Enable</b><table><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></tr><tr><td>0h</td><td>Flush Disabled</td><td>Depth relates caches (HiZ, Stencil and Depth) are NOT flushed.</td></tr><tr><td>1h</td><td>Flush Enabled</td><td>Depth relates caches (HiZ, Stencil and Depth) are flushed.</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 should be DISABLED for End-of-pipe (Read) fences, PS_DEPTH_COUNT or TIMESTAMP queries.</td></tr></table></td></tr>
<tr><td>2</td><td>31:2</td><td><b>Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:2]U32</td></tr></table>If <b>Post Sync Operation</b> is set to 1h (<b>LRI Post-Sync Operation</b> 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 If <b>LRI Post-Sync Operation</b> is set: Bits 22:2 (Bits 31:23 are reserved MBZ) specify the MMIO offset destination for the data in the <b>Immediate Data Low</b> (DW3) field. Only DW writes are valid.</td></tr>
<tr><td></td><td>1:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>3</td><td>31:0</td><td><b>Immediate Data</b><table><tr><td>Format:</td><td>U32</td></tr></table>This field specifies the Lower DWord value to be written to the targeted location. Only valid when <b>Post-Sync Operation</b> is 1h (Write Immediate Data) or <b>LRI Post-Sync Operation</b> is set. Ignored if Post-Sync Operation is "No write", "Write PS_DEPTH_COUNT" or "Write TIMESTAMP".<table><tr><th>Programming Notes</th></tr><tr><td>This field should be programmed to 0 when Post-Sync Operation is set to Write PS Depth Count or Write Timestamp.</td></tr></table></td></tr>
<tr><td>4</td><td>31:0</td><td><b>Immediate Data</b><table><tr><td>Format:</td><td>U32</td></tr></table>This field specifies the Upper DWord value to be written to the targeted location. Only valid when Post-Sync Operation is 1h (Write Immediate Data) Ignored if Post-Sync Operation is "No write", "Write PS_DEPTH_COUNT", "Write TIMESTAMP" or "LRI Post Sync Opeation".<table><tr><th>Programming Notes</th></tr><tr><td>This field should be programmed to 0 when Post-Sync Operation is set to Write PS</td></tr></table></td></tr>
</table>

PIPE_CONTROL		
		Depth Count or Write Timestamp.

PIPELINE_SELECT

Source:	BSpec
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)

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE
	28:27	**Command SubType** Default Value: 1h GFXPIPE_SINGLE_DW
	26:24	**3D Command Opcode** Value: 1h — Name: GFXPIPE_NONPIPELINED **[Default]**
	23:16	**3D Command Sub Opcode** Default Value: 04h GFXPIPE
	15:2	**Reserved**
	1:0	**Pipeline Select**

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

pln - Plane

Source:	EuIsa
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
Restriction: This is a specialized instruction that only supports an execution size (ExecSize) of 8 or 16.
Restriction: The src0 region must be a replicated scalar (with HorzStride == VertStride == 0).
Restriction: src0 must specify .0 or .4 as the subregister number, corresponding to a subregister byte offset of 0 or 16.
Restriction: Source operands cannot be accumulators.

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;
    }
}
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	N

Src Types	Dst Types

pln - Plane

F	F

DWord	Bit	Description
0..3	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM
	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

ret - Return

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

Restriction: This instruction cannot take accumulator as source.

Restriction: 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
Y	N	N	N

Src Types
D,UD

DWord	Bit	Description
0..3	127:64	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32
	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header**

ret - Return

		Format:	EU_INSTRUCTION_HEADER

rndd - Round Down

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

Restriction
Restriction: No accumulator access, implicit or explicit.

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	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32
	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

rnde - Round to Nearest or Even

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

Restriction
Restriction: No accumulator access, implicit or explicit.

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	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32
	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header**

rnde - Round to Nearest or Even

		Format:	EU_INSTRUCTION_HEADER

rndz - Round to Zero

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

Restriction
Restriction: No accumulator access, implicit or explicit.

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	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32
	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

rndu - Round Up

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

Restriction
Restriction: No accumulator access, implicit or explicit.

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	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32
	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

sel - Select

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).

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 a conditional modifier flushes any selected denorm source value to a zero destination value.

Format:
(pred) sel[.cmod] (exec_size) dst src0 src1

Restriction

Restriction: The maximum execution size is 16. SIMD32 is not supported.

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
*B,*W*D	*B,*W,*D
F	F
DF	DF

DWord	Bit	Description
0..3	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM
	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

send - Send Message

Source:	EuIsa
Length Bias:	4

Description
Send a message stored in GRF starting at <src> to a shared function identified by <ex_desc> along with control from <desc> with a GRF writeback location at <dest>.
The send instruction performs data communication between a thread and external function units, including shared functions (Sampler, Data Port Read, Data Port Write, URB, and Message Gateway) and some fixed functions (e.g. Thread Spawner, who also have an unique Shared Function ID). The send instruction adds an entry to the EU's message request queue. The request message is stored in a block of contiguous GRF registers. The response message, if present, will be returned to a block of contiguous GRF registers. The return GRF writes may be in any order depending on the external function units. <src> is the lead GRF register for request. <dest> is the lead GRF register for response. The message descriptor field <desc> contains the Message Length (the number of consecutive GRF registers) and the Response Length (the number of consecutive GRF registers). It also contains the header present bit, and the function control signals. The extend mesage descriptor field <ex_desc> contains the target function ID. WrEn is forwarded to the target function in the message sideband.
The send instruction is the only way to terminate a thread. When the EOT (End of Thread) bit of <ex_desc> is set, it indicates the end of thread to the EU, the Thread Dispatcher and, in most cases, the parent fixed function. Message descriptor field <desc> can be a 32-bit immediate, imm32, or a 32-bit scalar register, <reg32a>. GEN restricts that the 32-bit scalar register <reg32a> must be the leading dword of the address register. It should be in the form of a0.0<0;1,0>:ud. When <desc> is a register operand, only the lower 29 bits of <reg32a> are used.
<ex_desc> is a 6-bit immediate, imm6. The lower 4bits of the <ex_desc> specifies the SFID for the message. The MSb of the message descriptor, the EOT field, always comes from bit 127 of the instruction word, which is the MSb of imm6. A thread must terminate with a send instruction with EOT turned on.
<src> is a 256-bit aligned GRF register. It serves as the leading GRF register of the request.
<dest> serves for two purposes: to provide the leading GRF register location for the response message if present, and to provide parameters to form the channel enable sideband signals. <dest> signals whether there is a response to the message request. It can be either a null register, a direct-addressed GRF register or a register-indirect GRF register. Otherwise, hardware behavior is undefined. If <dest> is null, there is no response to the request. Meanwhile, the Response Length field in <desc> must be 0. Certain types of message requests, such as memory write (store) through the Data Port, do not want response data from the function unit. If so, the posted destination operand can be null. If <dest> is a GRF register, the register number is forwarded to the shared function. In this case, the target function unit must send one or more response message phases back to the requesting thread. The number of response message phases must match the Response Length field in <desc>, which of course cannot be zero. For some cases, it could be an empty return message. An empty return message is defined as a single phase message with all channel enables turned off. The subregister number, horizontal stride, destination mask and type fields of <dest> are always valid and are used in part to generate on the WrEn. This is true even if <dest> is a null register (this is an exception for null as for most cases these fields are ignored by hardware). The 16-bit channel enables of the message sideband are formed based on the WrEn. Interpretation of

send - Send Message

the channel enable sideband signals is subject to the target external function. In general for a 'send' instruction with return messages, they are used as the destination dword write mask for the GRF registers starting at <dest>. For a message that has multiple return phases, the same set of channel enable signals applies to all the return phases.

Thread managed memory coherency: A special usage of using non-null <dest> is to support write-commit signaling for memory write service by the Data Port Write unit. If <post_dest> is not null for a memory write request, the Data Port along with the Data Cache or Render Cache will wait until all the posted writes for the request have reached the coherent domain before sending back to the requesting thread an empty message to <dest> register. A memory write reaching the coherent domain, also referred to as reaching the global observable state, means that subsequent read to the same memory location, no matter which thread issues the read, must return the data of the write.

The destination dependency control, {NoDDClr}, can be used in this instruction. This allows software to control the destination dependencies for multiple 'read'-type messages similar to that for multiple instructions using EU execution pipeline. As send does not check register dependencies for the post destination, {NoDDChk} should not be used for this instruction.

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

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

DWord	Bit	Description
0..3	127:96	**Message** Format: EU_INSTRUCTION_OPERAND_SEND_MSG
	95:89	**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

send - Send Message

	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align16') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN16
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:28	**Controls B** Format: EU_INSTRUCTION_CONTROLS_B
	27:24	**Shared Function ID (SFID)** Format: SFID
	23:8	**Controls A** Format: EU_INSTRUCTION_CONTROLS_A
	7	**Reserved** Format: MBZ
	6:0	**Opcode** Format: EU_OPCODE

shl - Shift Left

Source:	EuIsa
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.
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.
Format: [(pred)] shl[.cmod] (exec_size) dst src0 src1

Restriction
Restriction: Accumulator cannot be destination, implicit or explicit.
Restriction: Results of saturation in packed-DWord mode are unpredicable.

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 = src1.chan[n] & 0x1F; // Always use low 5 bits for shift count.
        dst.chan[n] = src0.chan[n] << shiftCnt;
    }
}
```

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	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM
	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header**

shl - Shift Left

		Format:	EU_INSTRUCTION_HEADER

shr - Shift Right

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.

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.

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

```
Evaluate(WrEn); for ( n = 0; n < exec_size; n++ ) { if ( WrEn.chan[n] ) { shiftCnt
= src1.chan[n] & 0x1F; // Always use low 5 bits for shift count. dst.chan[n] =
src0.chan[n] >> shiftCnt; } }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types
UB,UW,UD	UB,UW,UD

DWord	Bit	Description
0..3	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM
	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

f32to16 - Single Precision Float to Half Precision Float

Source:	EuIsa
Length Bias:	4

The f32to16 instruction converts the single precision float in src0 to half precision float and storing in the lower word of each channel in dst.

Because this instruction does not have a 16-bit floating-point type, the destination data type must be Word (W).

Format:
[(pred)] f32to16[.cmod] (exec_size) dst src0

Restriction
Restriction: The destination must be DWord-aligned and specify a horizontal stride (HorzStride) of 2. The 16-bit result is stored in the lower word of each destination channel and the upper word is not modified.
Restriction: The FP Mode (Single Precision Floating Point Mode in cr0) must be IEEE mode.
Restriction: No accumulator access, implicit or explicit.

Syntax

```
[(pred)] f32to16[.cmod] (exec_size) reg reg [(pred)] f32to16[.cmod] (exec_size) reg imm32
```

Pseudocode

```
Evaluate(WrEn); for ( n = 0; n < exec_size; n++ ) { if ( WrEn.chan[n] ) { dst.chan[n] =
convert single precision float to half precision float(src0.chan[n]); } }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types
F	W

DWord	Bit	Description
0..3	127:64	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32
	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

SRC_COPY_BLT

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. The command must indicate the horizontal and vertical directions: either forward or backwards to avoid data corruption. The X direction (horizontal) field applies to both the destination and source operands. The source and destination pitches (stride) are signed.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="5">0
BR00</td><td>31:29</td><td><b>Client</b><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><b>Instruction Target(Opcode)</b><table><tr><td>Default Value:</td><td>43h</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td>21:20</td><td><b>32bpp Byte Mask</b>
This field is only used for 32bpp.<table><tr><th>Value</th><th>Name</th></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>19:6</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>5:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>04h</td></tr></table></td></tr>
<tr><td rowspan="5">1
BR13</td><td>31</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>30</td><td><b>X Direction</b>
(1 = written from right to left (decrementing = backwards); 0 = incrementing)</td></tr>
<tr><td>29:26</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>25:24</td><td><b>Color Depth</b><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><b>Raster Operation</b></td></tr>
</table>

SRC_COPY_BLT

	15:0	**Destination Pitch (signed)** Destination pitch in bytes (Same as before).
2 BR14	31:16	**Destination Height (in scan lines)**
	15:0	**Destination Byte Width (in bytes)**
3 BR09	31:0	**Destination Address** Format: | GraphicsAddress[31:0] Address of the first byte to be written.
4 BR11	31:16	**Reserved** Format: | MBZ
	15:0	**Source Pitch** (double word aligned and signed)
5 BR12	31:0	**Source Address** Format: | GraphicsAddress[31:0] Address of the first byte to be read.

STATE_BASE_ADDRESS

Source:	BSpec
Length Bias:	2

The STATE_BASE_ADDRESS command sets the base pointers for subsequent state, instruction, and media indirect object accesses by the GPE. (See Table 4-3. Base Address Utilization for details)

Programming Notes

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

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h GFXPIPE</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Command SubType</b><table><tr><td>Default Value:</td><td>0h GFXPIPE_COMMON</td></tr></table></td></tr>
<tr><td></td><td>26:24</td><td><b>3D Command Opcode</b><table><tr><td>Default Value:</td><td>1h GFXPIPE_NONPIPELINED</td></tr></table></td></tr>
<tr><td></td><td>23:16</td><td><b>3D Command Sub Opcode</b><table><tr><td>Default Value:</td><td>01h STATE_BASE_ADDRESS</td></tr></table></td></tr>
<tr><td></td><td>15:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><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>8h</td><td>DWORD_COUNT_n <b>[Default]</b></td><td>Excludes DWord (0,1)</td></tr></table></td></tr>
<tr><td>1</td><td>31:12</td><td><b>General State Base Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:12]</td></tr></table>Specifies the 4K-byte aligned base address for general state accesses. See Table 4-3 for details on where this base address is used.</td></tr>
<tr><td></td><td>11:8</td><td><b>General State Memory Object Control State</b><table><tr><td>Format:</td><td>MEMORY_OBJECT_CONTROL_STATE</td></tr></table>Specifies the memory object control state for indirect state using the General State Base Address, with the exception of the stateless data port accesses.</td></tr>
</table>

<table>
<tr><th colspan="3">STATE_BASE_ADDRESS</th></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td>7:4</td><td><b>Stateless Data Port Access Memory Object Control State</b>
<table><tr><td>Format:</td><td>MEMORY_OBJECT_CONTROL_STATE</td></tr></table>
Specifies the memory object control state for stateless data port accesses.</td></tr>
<tr><td></td><td>3</td><td><b>Stateless Data Port Access Force Write Thru</b>
<table><tr><td>Format:</td><td>U1</td></tr></table>
0: If the stateless data port access memory object control indicates L3 cachable the accesses will be write back cacheable.
1: If the stateless data port access memory object control indicates L3 cachable the accesses will be write thru cacheable.</td></tr>
<tr><td></td><td>2:1</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>0</td><td><b>General State Base Address Modify Enable</b>
<table><tr><td>Format:</td><td>Enable</td></tr></table>
The other fields in this dword are updated only when this bit is set.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Disable</td><td>Ignore the updated address</td></tr>
<tr><td>1h</td><td>Enable</td><td>Modify the address</td></tr>
</table></td></tr>
<tr><td>2</td><td>31:12</td><td><b>Surface State Base Address</b>
<table><tr><td>Format:</td><td>GraphicsAddress[31:12]</td></tr></table>
Specifies the 4K-byte aligned base address for binding table and surface state accesses. See Table 4-3 for details on where this base address is used.</td></tr>
<tr><td></td><td>11:8</td><td><b>Surface State Memory Object Control State</b>
<table><tr><td>Format:</td><td>MEMORY_OBJECT_CONTROL_STATE</td></tr></table>
Specifies the memory object control state for indirect state using the <b>Surface State Base Address</b>.</td></tr>
<tr><td></td><td>7:1</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>0</td><td><b>Surface State Base Address Modify Enable</b>
<table><tr><td>Format:</td><td>Enable</td></tr></table>
The other fields in this dword are updated only when this bit is set.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Disable</td><td>Ignore the updated address</td></tr>
<tr><td>1h</td><td>Enable</td><td>Modify the address</td></tr>
</table></td></tr>
<tr><td>3</td><td>31:12</td><td><b>Dynamic State Base Address</b>
<table><tr><td>Format:</td><td>GraphicsAddress[31:12]</td></tr></table></td></tr>
</table>

STATE_BASE_ADDRESS

<table>
<tr><td></td><td></td><td>Specifies the 4K-byte aligned base address for sampler and viewport state accesses. See Table 4-3 for details on where this base address is used.</td></tr>
<tr><td></td><td>11:8</td><td><b>Dynamic State Memory Object Control State</b><table><tr><td>Format:</td><td>MEMORY_OBJECT_CONTROL_STATE</td></tr></table>Specifies the memory object control state for indirect state using the <b>Dynamic State Base Address</b>. 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.</td></tr>
<tr><td></td><td>7:1</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>0</td><td><b>Dynamic State Base Address Modify Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>The other fields in this dword are updated only when this bit is set.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Disable</td><td>Ignore the updated address</td></tr><tr><td>1h</td><td>Enable</td><td>Modify the address</td></tr></table></td></tr>
<tr><td>4</td><td>31:12</td><td><b>Indirect Object Base Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:12]</td></tr></table>Specifies the 4K-byte aligned base address for indirect object load in MEDIA_OBJECT command. See Table 4-3 for details on where this base address is used.</td></tr>
<tr><td></td><td>11:8</td><td><b>Indirect Object Memory Object Control State</b><table><tr><td>Format:</td><td>MEMORY_OBJECT_CONTROL_STATE</td></tr></table>Specifies the memory object control state for indirect objects using the <b>Indirect Object Base Address</b>.</td></tr>
<tr><td></td><td>7:1</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>0</td><td><b>Indirect Object Base Address Modify Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>The other fields in this dword are updated only when this bit is set.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Disable</td><td>Ignore the updated address</td></tr><tr><td>1h</td><td>Enable</td><td>Modify the address</td></tr></table></td></tr>
<tr><td>5</td><td>31:12</td><td><b>Instruction Base Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:12]</td></tr></table>Specifies the 4K-byte aligned base address for all EU instruction accesses.</td></tr>
<tr><td></td><td>11:8</td><td><b>Instruction Memory Object Control State</b></td></tr>
</table>

<table>
<tr><th colspan="3">STATE_BASE_ADDRESS</th></tr>
<tr><td></td><td></td><td><table><tr><td>Format:</td><td>MEMORY_OBJECT_CONTROL_STATE</td></tr></table>Specifies the memory object control state for EU instructions using the <b>Instruction Base Address</b>.</td></tr>
<tr><td></td><td>7:1</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>0</td><td><b>Instruction Base Address Modify Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>The other fields in this dword are updated only when this bit is set.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Disable</td><td>Ignore the updated address</td></tr><tr><td>1h</td><td>Enable</td><td>Modify the address</td></tr></table></td></tr>
<tr><td>6</td><td>31:12</td><td><b>General State Access Upper Bound</b><table><tr><td>Format:</td><td>GraphicsAddress[31:12]</td></tr></table>Specifies the 4K-byte aligned (exclusive) maximum Graphics Memory address for general state accesses. This includes all accesses that are offset from <b>General State Base Address</b> (see Table 4-3). Read accesses from this address and beyond will return UNDEFINED values. Data port writes to this address and beyond will be "dropped on the floor" (all data channels will be disabled so no writes occur). Setting this field to 0 will cause this range check to be ignored.

If non-zero, this address must be greater than the General State Base Address.</td></tr>
<tr><td></td><td>11:1</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>0</td><td><b>General State Access Upper Bound Modify Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>The bound in this dword is updated only when this bit is set.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Disable</td><td>Ignore the updated bound</td></tr><tr><td>1h</td><td>Enable</td><td>Modify the bound</td></tr></table></td></tr>
<tr><td>7</td><td>31:12</td><td><b>Dynamic State Access Upper Bound</b><table><tr><td>Format:</td><td>GraphicsAddress[31:12]</td></tr></table>Specifies the 4K-byte aligned (exclusive) maximum Graphics Memory address for dynamic state accesses. This includes all accesses that are offset from <b>Dynamic State Base Address</b> (see Table 4-3). Read accesses from this address and beyond will return UNDEFINED values. Data port writes to this address and beyond will be "dropped on the floor" (all data channels will be disabled so no writes occur). Setting this field to 0 will cause this range check to be ignored.</td></tr>
</table>

<table>
<tr><th colspan="3">STATE_BASE_ADDRESS</th></tr>
<tr><td></td><td></td><td>If non-zero, this address must be greater than the <b>Dynamic State Base Address</b>.</td></tr>
<tr><td></td><td>11:1</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>0</td><td><b>Dynamic State Access Upper Bound Modify Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>The bound in this dword is updated only when this bit is set.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Disable</td><td>Ignore the updated bound</td></tr><tr><td>1h</td><td>Enable</td><td>Modify the bound</td></tr></table></td></tr>
<tr><td>8</td><td>31:12</td><td><b>Indirect Object Access Upper Bound</b><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 an indirect object load in a MEDIA_OBJECT command. Indirect data accessed at this address and beyond will appear to be 0. Setting this field to 0 will cause this range check to be ignored.
If non-zero, this address must be greater than the <b>Indirect Object Base Address</b>.
Hardware ignores this field if indirect data is not present.
Setting this field to FFFFFh will cause this range check to be ignored.</td></tr>
<tr><td></td><td>11:1</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>0</td><td><b>Indirect Object Access Upper Bound Modify Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>The bound in this dword is updated only when this bit is set.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Disable</td><td>Ignore the updated bound</td></tr><tr><td>1h</td><td>Enable</td><td>Modify the bound</td></tr></table></td></tr>
<tr><td>9</td><td>31:12</td><td><b>Instruction Access Upper Bound</b><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 an EU instruction. Instruction data accessed at this address and beyond will return UNDEFINED values. Setting this field to 0 will cause this range check to be ignored.
If non-zero, this address must be greater than the <b>Instruction Base Address</b>.</td></tr>
<tr><td></td><td>11:1</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>0</td><td><b>Instruction Access Upper Bound Modify Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table></td></tr>
</table>

<table>
<tr><th colspan="3">STATE_BASE_ADDRESS</th></tr>
<tr><td></td><td></td><td>The bound in this dword is updated only when this bit is set.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Disable</td><td>Ignore the updated bound</td></tr>
<tr><td>1h</td><td>Enable</td><td>Modify the bound</td></tr>
</table>
</td></tr>
</table>

STATE_SIP

Source:	BSpec
Length Bias:	2

The STATE_SIP command specifies the starting instruction location of the System Routine that is shared by all threads in execution.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h GFXPIPE</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Command SubType</b><table><tr><td>Default Value:</td><td>0h GFXPIPE_COMMON</td></tr></table></td></tr>
<tr><td></td><td>26:24</td><td><b>3D Command Opcode</b><table><tr><td>Default Value:</td><td>1h GFXPIPE_NONPIPELINED</td></tr></table></td></tr>
<tr><td></td><td>23:16</td><td><b>3D Command Sub Opcode</b><table><tr><td>Default Value:</td><td>02h STATE_SIP</td></tr></table></td></tr>
<tr><td></td><td>15:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><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>0h</td><td>DWORD_COUNT_n <b>[Default]</b></td><td>Excludes DWord (0,1)</td></tr></table></td></tr>
<tr><td>1</td><td>31:4</td><td><b>System Instruction Pointer</b><table><tr><td>Format:</td><td>InstructionBaseOffset[31:4]Kernel</td></tr></table>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.</td></tr>
<tr><td></td><td>3:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
</table>

sad2 - Sum of Absolute Difference 2

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
Restriction: Source operands cannot be accumulators.
Restriction: 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	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM
	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

sada2 - Sum of Absolute Difference Accumulate 2

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
Restriction: Source operands cannot be accumulators.
Restriction: 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
Y	Y	Y	Y

Src Types	Dst Types
B,UB	W,UW

DWord	Bit	Description
0..3	127:64	**ImmSource** Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_REG_IMM
	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS

sada2 - Sum of Absolute Difference Accumulate 2

	31:0	**Header** Format: EU_INSTRUCTION_HEADER

SWTESS_BASE_ADDRESS

Source:	BSpec
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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>3h GFXPIPE</td></tr></table></td></tr>
<tr><td></td><td>28:27</td><td><b>Command SubType</b><table><tr><td>Default Value:</td><td>0h GFXPIPE_COMMON</td></tr></table></td></tr>
<tr><td></td><td>26:24</td><td><b>3D Command Opcode</b><table><tr><td>Default Value:</td><td>1h GFXPIPE_NONPIPELINED</td></tr></table></td></tr>
<tr><td></td><td>23:16</td><td><b>3D Command Sub Opcode</b><table><tr><td>Default Value:</td><td>03h SWTESS_BASE_ADDRESS</td></tr></table></td></tr>
<tr><td></td><td>15:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><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>0h</td><td>DWORD_COUNT_n <b>[Default]</b></td><td>Excludes DWord (0,1)</td></tr></table></td></tr>
<tr><td>1</td><td>31:12</td><td><b>SW Tessellation Base Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:12]</td></tr></table>Specifies the 4K-byte aligned base address for TE unit SW tessellation data read accesses.</td></tr>
<tr><td></td><td>11:8</td><td><b>SW Tessellation Memory Object Control State</b><table><tr><td>Format:</td><td>MEMORY_OBJECT_CONTROL_STATE</td></tr></table>Specifies the memory object control state used by the TE unit to read SW tessellation data from memory.</td></tr>
<tr><td></td><td>7:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
</table>

wait - Wait Notification

Source:	EuIsa
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:ud is for thread to thread communication (via the Message Gateway shared function) and n1:ud for host to thread communication (through MMIO registers). See the Message Gateway chapter for thread-thread communication.

Format:
wait (exec_size) nreg

Restriction
Restriction: src0 and dst must be n0, n1, or n2.
Restriction: Execution size must be 1 as the notification registers are scalar.
Restriction: Predication is not allowed.
Restriction: 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_IMM32
	127:64	**Sources**

wait - Wait Notification

		Exists If: ([Operand Control][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

while - While

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.ld 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 16-bit jump target offset JIP. JIP is a signed 16-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 W (signed word integer).
Format: [(pred)] while (exec_size) JIP

Restriction
Restriction: The execution size must be the same for the while instruction and any break and cont instructions of the same code block.

Syntax

```
[(pred)] while (exec_size) imm16
```

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	127:112	**Reserved** Format: MBZ
	111:96	**JIP** Format: S15 Jump Target Offset. The relative offset in 64-bit units if a jump is taken for the instruction.
	95:91	**Reserved**

while - While

		Format: MBZ
	90	**Flag Register Number** Added a second flag register
	89	**Flag Subregister Number** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN1
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align16') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN16
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

XY_COLOR_BLT

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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="7">0
BR00</td><td>31:29</td><td><b>Client</b>
<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><b>Instruction Target(Opcode)</b>
<table><tr><td>Default Value:</td><td>50h</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td>21:20</td><td><b>32bpp Byte Mask</b>
This field is only used for 32bpp.
<table><tr><th>Value</th><th>Name</th></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>19:12</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>11</td><td><b>Tiling Enable</b>
<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>Tile-X or Tile-Y.</td></tr></table></td></tr>
<tr><td>10:8</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>7:0</td><td><b>DWord Length</b>
<table><tr><td>Default Value:</td><td>04h</td></tr></table></td></tr>
<tr><td rowspan="2">1
BR13</td><td>31</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>30</td><td><b>Clipping Enabled</b>
<table><tr><th>Value</th><th>Name</th></tr><tr><td>0b</td><td>Disabled</td></tr></table></td></tr>
</table>

XY_COLOR_BLT

<table>
<tr><td></td><td></td><td><table><tr><td>1b</td><td>Enabled</td></tr></table></td></tr>
<tr><td></td><td>29:26</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>25:24</td><td><b>Color Depth</b><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><b>Raster Operation</b></td></tr>
<tr><td></td><td>15:0</td><td><b>Destination Pitch in DWords</b>
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><b>Destination Y1 Coordinate (Top)</b>
16 bit signed number.</td></tr>
<tr><td></td><td>15:0</td><td><b>Destination X1 Coordinate (Left)</b>
16 bit signed number.</td></tr>
<tr><td>3
BR23</td><td>31:16</td><td><b>Destination Y2 Coordinate (Bottom)</b>
16 bit signed number.</td></tr>
<tr><td></td><td>15:0</td><td><b>Destination X2 Coordinate (Right)</b>
16 bit signed number.</td></tr>
<tr><td>4
BR09</td><td>31:0</td><td><b>Setup Destination Base Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr></table>Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes.</td></tr>
<tr><td>5
BR16</td><td>31:0</td><td><b>Solid Pattern Color</b>
8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]</td></tr>
</table>

XY_FULL_BLT

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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0
BR00</td><td>31:29</td><td><b>Client</b>
<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></td><td>28:22</td><td><b>Instruction Target(Opcode)</b>
<table>
<tr><td>Default Value:</td><td>55h</td></tr>
<tr><td>Format:</td><td>Opcode</td></tr>
</table></td></tr>
<tr><td></td><td>21:20</td><td><b>32bpp Byte Mask</b>
This field is only used for 32bpp.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>00b</td><td><b>[Default]</b></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></td><td>19:16</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>15</td><td><b>Src Tiling Enable</b></td></tr>
</table>

XY_FULL_BLT

<table>
<tr><td rowspan="5"></td><td></td><td><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>Tile-X or Tile-Y.</td></tr></table></td></tr>
<tr><td>14:12</td><td><b>Pattern Horizontal Seed</b>
Pixel of the scan line to start on corresponding to DST X=0.</td></tr>
<tr><td>11</td><td><b>Dest Tiling Enable</b><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>Tile-X or Tile-Y.</td></tr></table></td></tr>
<tr><td>10:8</td><td><b>Pattern Vertical Seed</b>
Starting scan line of the 8x8 pattern corresponding to DST Y=0.</td></tr>
<tr><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>07h</td></tr></table></td></tr>
<tr><td rowspan="7">1
BR13</td><td>31</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>30</td><td><b>Clipping Enabled</b><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>29:26</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>25:24</td><td><b>Color Depth</b><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><b>Raster Operation</b></td></tr>
<tr><td>15:0</td><td><b>Destination Pitch in DWords</b>
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><b>Destination Y1 Coordinate (Top)</b>
16 bit signed number.</td></tr>
<tr><td>15:0</td><td><b>Destination X1 Coordinate (Left)</b>
16 bit signed number.</td></tr>
<tr><td rowspan="2">3
BR23</td><td>31:16</td><td><b>Destination Y2 Coordinate (Bottom)</b>
16 bit signed number.</td></tr>
<tr><td>15:0</td><td><b>Destination X2 Coordinate (Right)</b>
16 bit signed number.</td></tr>
</table>

XY_FULL_BLT		
4 BR09	31:0	**Destination Base Address** Format: | GraphicsAddress[31:0] Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes.
5 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).
6 BR26	31:16	**Source Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Source X1 Coordinate (Left)** 16 bit signed number.
7 BR12	31:0	**Source Address** Format: | GraphicsAddress[31:0] Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_15 enabled), this address is limited to 4Kbytes.
8 BR15	31:0	**Pattern Base** (28:06 are implemented) (Note no NPO2 change here). The pattern data must be located in linear memory.

XY_FULL_IMMEDIATE_PATTERN_BLT

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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="4">0
BR00</td><td>31:29</td><td><b>Client</b>
<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><b>Instruction Target(Opcode)</b>
<table>
<tr><td>Default Value:</td><td>74h</td></tr>
<tr><td>Format:</td><td>Opcode</td></tr>
</table></td></tr>
<tr><td>21:20</td><td><b>32bpp Byte Mask</b>
This field is only used for 32bpp.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>00b</td><td><b>[Default]</b></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><b>Reserved</b></td></tr>
</table>

<table>
<tr><th colspan="3">XY_FULL_IMMEDIATE_PATTERN_BLT</th></tr>
<tr><td></td><td></td><td><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>15</td><td><b>Src Tiling Enable</b><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>Tile-X or Tile-Y.</td></tr></table></td></tr>
<tr><td></td><td>14:12</td><td><b>Pattern Horizontal Seed</b>
(pixel of the scan line to start on corresponding to DST X=0)</td></tr>
<tr><td></td><td>11</td><td><b>Dest Tiling Enable</b><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>Tile-X or Tile-Y.</td></tr></table></td></tr>
<tr><td></td><td>10:8</td><td><b>Pattern Vertical Seed</b>
Starting scan line of the 8x8 pattern corresponding to DST Y=0.</td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>06h Excludes DWORD 0,1</td></tr></table>06 + DWL = (Number of Immediate double words)h</td></tr>
<tr><td>1
BR13</td><td>31</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>30</td><td><b>Clipping Enabled</b><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>29:26</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>25:24</td><td><b>Color Depth</b><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><b>Raster Operation</b></td></tr>
<tr><td></td><td>15:0</td><td><b>Destination Pitch in DWords</b>
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><b>Destination Y1 Coordinate (Top)</b>
16 bit signed number.</td></tr>
<tr><td></td><td>15:0</td><td><b>Destination X1 Coordinate (Left)</b>
16 bit signed number.</td></tr>
</table>

XY_FULL_IMMEDIATE_PATTERN_BLT

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. When Src Tiling is enabled (Bit_15 enabled), this address is limited to 4Kbytes.
5 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 11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
6 BR26	31:16	**Source Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Source X1 Coordinate (Left)** 16 bit signed number.
7 BR12	31:0	**Source Address** Format: GraphicsAddress[31:0] Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit 15 enabled), this address is limited to 4Kbytes.
8..n	31:0	**Immediate Data 0**

XY_FULL_MONO_PATTERN_BLT

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.

<table>
<tr><th colspan="3">XY_FULL_MONO_PATTERN_BLT</th></tr>
<tr><td rowspan="7"></td><td></td><td><table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td><b>[Default]</b></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><b>Reserved</b></td></tr>
<tr><td>15</td><td><b>Src Tiling Enable</b><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>Tile-X or Tile-Y.</td></tr></table></td></tr>
<tr><td>14:12</td><td><b>Pattern Horizontal Seed</b>
(pixel of the scan line to start on corresponding to DST X=0)</td></tr>
<tr><td>11</td><td><b>Dest Tiling Enable</b><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>Tile-X or Tile-Y.</td></tr></table></td></tr>
<tr><td>10:8</td><td><b>Pattern Vectical Seed</b>
Starting scan line of the 8x8 pattern corresponding to DST Y=0.</td></tr>
<tr><td>7:0</td><td><b>DWord Length</b><table><tr><th>Value</th><th>Name</th></tr><tr><td>0Ah</td><td></td></tr></table></td></tr>
<tr><td rowspan="7">1
BR13</td><td>31</td><td><b>Solid Pattern Select</b><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><b>Clipping Enabled</b><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><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>28:27</td><td><b>Mono Source Transparency Mode</b><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><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>25:24</td><td><b>Color Depth</b></td></tr>
</table>

XY_FULL_MONO_PATTERN_BLT

<table>
<tr><td></td><td></td><td><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><b>Raster Operation</b></td></tr>
<tr><td></td><td>15:0</td><td><b>Destination Pitch in DWords</b>
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><b>Destination Y1 Coordinate (Top)</b>
16 bit signed number.</td></tr>
<tr><td>15:0</td><td><b>Destination X1 Coordinate (Left)</b>
16 bit signed number.</td></tr>
<tr><td rowspan="2">3
BR23</td><td>31:16</td><td><b>Destination Y2 Coordinate (Bottom)</b>
16 bit signed number.</td></tr>
<tr><td>15:0</td><td><b>Destination X2 Coordinate (Right)</b>
16 bit signed number.</td></tr>
<tr><td>4
BR09</td><td>31:0</td><td><b>Destination Base Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr></table>Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes.</td></tr>
<tr><td rowspan="2">5
BR11</td><td>31:16</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>15:0</td><td><b>Source Pitch (double word aligned and signed) and in DWords</b>
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).</td></tr>
<tr><td rowspan="2">6
BR26</td><td>31:16</td><td><b>Source Y1 Coordinate (Top)</b>
16 bit signed number.</td></tr>
<tr><td>15:0</td><td><b>Source X1 Coordinate (Left)</b>
16 bit signed number.</td></tr>
<tr><td>7
BR12</td><td>31:0</td><td><b>Source Base Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr></table>(base address of the source surface: X=0, Y=0). When Src Tiling is enabled (Bit 15 enabled), this address is limited to 4Kbytes.</td></tr>
<tr><td>8
BR16</td><td>31:0</td><td><b>Pattern Background Color</b>
8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]</td></tr>
<tr><td>9
BR17</td><td>31:0</td><td><b>Pattern Foreground Color</b>
8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]</td></tr>
</table>

XY_FULL_MONO_PATTERN_BLT

10 BR20	31:0	**Pattern Data 0** (least significant DW)
11 BR21	31:0	**Pattern Data 1** (most significant DW)

XY_FULL_MONO_PATTERN_MONO_SRC_BLT

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.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)**

XY_FULL_MONO_PATTERN_MONO_SRC_BLT

<table>
<tr><td></td><td></td><td><table><tr><td>Default Value:</td><td>58h</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td></td><td>21:20</td><td><b>32bpp Byte Mask</b>
This field is only used for 32bpp.<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td><b>[Default]</b></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></td><td>19:17</td><td><b>Monochrome source data bit position of the first pixel within a byte per scan line.</b></td></tr>
<tr><td></td><td>16:15</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>14:12</td><td><b>Pattern Horizontal Seed</b>
(pixel of the scan line to start on corresponding to DST X=0)</td></tr>
<tr><td></td><td>11</td><td><b>Tiling Enable</b><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>Tile-X or Tile-Y.</td></tr></table></td></tr>
<tr><td></td><td>10:8</td><td><b>Pattern Vertical Seed</b>
Starting scan line of the 8x8 pattern corresponding to DST Y = 0.</td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><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><b>Solid Pattern Select</b><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><b>Clipping Enabled</b><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><b>Mono Source Transparency Mode</b><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><b>Mono Pattern Transparency Mode</b><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>
</table>

XY_FULL_MONO_PATTERN_MONO_SRC_BLT

<table>
<tr><td></td><td>27:26</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>25:24</td><td><b>Color Depth</b><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><b>Raster Operation</b></td></tr>
<tr><td></td><td>15:0</td><td><b>Destination Pitch in DWords</b>
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><b>Destination Y1 Coordinate (Top)</b>
16 bit signed number.</td></tr>
<tr><td></td><td>15:0</td><td><b>Destination X1 Coordinate (Left)</b>
16 bit signed number.</td></tr>
<tr><td>3
BR23</td><td>31:16</td><td><b>Destination Y2 Coordinate (Bottom)</b>
16 bit signed number.</td></tr>
<tr><td></td><td>15:0</td><td><b>Destination X2 Coordinate (Right)</b>
16 bit signed number.</td></tr>
<tr><td>4
BR09</td><td>31:0</td><td><b>Destination Base Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr></table>Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes.</td></tr>
<tr><td>5
BR12</td><td>31:0</td><td><b>Mono Source Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr></table>(address corresponds to DST X1, Y1) (Note no NPO2 change here).</td></tr>
<tr><td>6
BR18</td><td>31:0</td><td><b>Source Background Color</b>
8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]</td></tr>
<tr><td>7
BR19</td><td>31:0</td><td><b>Source Foreground Color</b>
8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]</td></tr>
<tr><td>8
BR16</td><td>31:0</td><td><b>Pattern Background Color</b>
8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]</td></tr>
<tr><td>9
BR17</td><td>31:0</td><td><b>Pattern Foreground Color</b>
8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]</td></tr>
<tr><td>10</td><td>31:0</td><td><b>Pattern Data 0</b></td></tr>
</table>

XY_FULL_MONO_PATTERN_MONO_SRC_BLT

BR20		(least significant DW)
11 BR21	31:0	**Pattern Data 1** (most significant DW)

XY_FULL_MONO_SRC_BLT

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="3">0
BR00</td><td>31:29</td><td><b>Client</b>
<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><b>Instruction Target(Opcode)</b>
<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><b>32bpp Byte Mask</b>
This field is only used for 32bpp.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>00b</td><td><b>[Default]</b></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>
</table>

XY_FULL_MONO_SRC_BLT

<table>
<tr><td></td><td>19:17</td><td><b>Monochrome source data bit position of the first pixel within a byte per scan line.</b></td></tr>
<tr><td></td><td>16:15</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>14:12</td><td><b>Pattern Horizontal Seed</b>
(pixel of the scan line to start on corresponding to DST X=0)</td></tr>
<tr><td></td><td>11</td><td><b>Tiling Enable</b><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>Tile-X or Tile-Y.</td></tr></table></td></tr>
<tr><td></td><td>10:8</td><td><b>Pattern Vertical Seed</b>
Starting scan line of the 8x8 pattern corresponding to DST Y = 0.</td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><th>Value</th><th>Name</th></tr><tr><td>07h</td><td></td></tr></table></td></tr>
<tr><td>1
BR13</td><td>31</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>30</td><td><b>Clipping Enabled</b><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><b>Mono Source Transparency Mode</b><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><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>25:24</td><td><b>Color Depth</b><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><b>Raster Operation</b></td></tr>
<tr><td></td><td>15:0</td><td><b>Destination Pitch in DWords</b>
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</td><td>31:16</td><td><b>Destination Y1 Coordinate (Top)</b>
16 bit signed number.</td></tr>
</table>

XY_FULL_MONO_SRC_BLT		
BR22	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. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes.
5 BR12	31:0	**Mono Source Address** Format: GraphicsAddress[31:0] (address corresponds to DST X1, Y1) (Note no NPO2 change here).
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 BR15	31:0	**Pattern Base Address** Format: GraphicsAddress[31:0] (28:06 are implemented) (Note no NPO2 change here). The pattern data must be located in linear memory.

XY_FULL_MONO_SRC_IMMEDIATE_PATTERN_BLT

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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="3">0
BR00</td><td>31:29</td><td><b>Client</b>
<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><b>Instruction Target(Opcode)</b>
<table>
<tr><td>Default Value:</td><td>75h</td></tr>
<tr><td>Format:</td><td>Opcode</td></tr>
</table></td></tr>
<tr><td>21:20</td><td><b>32bpp Byte Mask</b>
This field is only used for 32bpp.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>00b</td><td><b>[Default]</b></td></tr>
<tr><td>1xb</td><td>Write Alpha Channel</td></tr>
</table></td></tr>
</table>

XY_FULL_MONO_SRC_IMMEDIATE_PATTERN_BLT

<table>
<tr><td></td><td></td><td><table><tr><td>x1b</td><td>Write RGB Channel</td></tr></table></td></tr>
<tr><td></td><td>19:17</td><td><b>Monochrome source data bit position of the first pixel within a byte per scan line.</b></td></tr>
<tr><td></td><td>16:15</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>14:12</td><td><b>Pattern Horizontal Seed</b>
(pixel of the scan line to start on corresponding to DST X=0)</td></tr>
<tr><td></td><td>11</td><td><b>Tiling Enable</b><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>Tile-X or Tile-Y.</td></tr></table></td></tr>
<tr><td></td><td>10:8</td><td><b>Pattern Vertical Seed</b>
Starting scan line of the 8x8 pattern corresponding to DST Y=0.</td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>06h Excludes DWORD 0,1</td></tr></table>06 + DWL = (Number of Immediate double words)h</td></tr>
<tr><td>1
BR13</td><td>31</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>30</td><td><b>Clipping Enabled</b><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><b>Mono Source Transparency Mode</b><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><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>25:24</td><td><b>Color Depth</b><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><b>Raster Operation</b></td></tr>
<tr><td></td><td>15:0</td><td><b>Destination Pitch in DWords</b>
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>
</table>

XY_FULL_MONO_SRC_IMMEDIATE_PATTERN_BLT

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. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes.
5 BR12	31:0	**Mono Source Address** Format: GraphicsAddress[31:0] (address corresponds to DST X1, Y1) (Note no NPO2 change here).
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_MONO_PAT_BLT

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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0
BR00</td><td>31:29</td><td><b>Client</b>
<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></td><td>28:22</td><td><b>Instruction Target(Opcode)</b>
<table>
<tr><td>Default Value:</td><td>52h</td></tr>
<tr><td>Format:</td><td>Opcode</td></tr>
</table></td></tr>
<tr><td></td><td>21:20</td><td><b>32bpp Byte Mask</b>
This field is only used for 32bpp.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>00b</td><td><b>[Default]</b></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></td><td>19:15</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>14:12</td><td><b>Pattern Horizontal Seed</b>
Pixel of the scan line to start on corresponding to DST X=0.</td></tr>
<tr><td></td><td>11</td><td><b>Tiling Enable</b>
<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>Tile-X or Tile-Y.</td></tr>
</table></td></tr>
</table>

XY_MONO_PAT_BLT

<table>
<tr><td></td><td>10:8</td><td><b>Pattern Vertical Seed</b>
Scan line of the 8x8 pattern to start on corresponding to DST Y=0.</td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><th>Value</th><th>Name</th></tr><tr><td>07h</td><td></td></tr></table></td></tr>
<tr><td>1
BR13</td><td>31</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>30</td><td><b>Clipping Enabled</b><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><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>28</td><td><b>Mono Pattern Transparency Mode</b><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><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>25:24</td><td><b>Color Depth</b><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><b>Raster Operation</b></td></tr>
<tr><td></td><td>15:0</td><td><b>Destination Pitch in DWords</b>
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><b>Destination Y1 Coordinate (Top)</b>
16 bit signed number.</td></tr>
<tr><td></td><td>15:0</td><td><b>Destination X1 Coordinate (Left)</b>
16 bit signed number.</td></tr>
<tr><td>3
BR23</td><td>31:16</td><td><b>Destination Y2 Coordinate (Bottom)</b>
16 bit signed number.</td></tr>
<tr><td></td><td>15:0</td><td><b>Destination X2 Coordinate (Right)</b>
16 bit signed number.</td></tr>
<tr><td>4</td><td>31:0</td><td><b>Destination Base Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr></table></td></tr>
</table>

XY_MONO_PAT_BLT

BR09		Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit 11 enabled), this address is limited to 4Kbytes.
5 BR16	31:0	**Pattern Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
6 BR17	31:0	**Pattern Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
7 BR20	31:0	**Pattern Data 0**
8 BR21	31:0	**Pattern Data 1**

XY_MONO_PAT_FIXED_BLT

Source:	BlitterCS
Length Bias:	2

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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="5">0
BR00</td><td>31:29</td><td><b>Client</b>
<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><b>Instruction Target(Opcode)</b>
<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><b>32bpp Byte Mask</b>
This field is only used for 32bpp.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>00b</td><td><b>[Default]</b></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><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td>18:15</td><td><b>Fixed Pattern</b>
<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>
</table></td></tr>
</table>

XY_MONO_PAT_FIXED_BLT

<table>
<tr><td></td><td></td><td><table>
<tr><td>0010b</td><td>HS_FDIAGONAL</td></tr>
<tr><td>0011b</td><td>HS_BDIAGONAL</td></tr>
<tr><td>0100b</td><td>HS_CROSS</td></tr>
<tr><td>0101b</td><td>HS_DIAGCROSS</td></tr>
<tr><td>0110b</td><td>Reserved</td></tr>
<tr><td>0111b</td><td>Reserved</td></tr>
<tr><td>1000b</td><td>Screen Door</td></tr>
<tr><td>1001b</td><td>SD Wide</td></tr>
<tr><td>1010b</td><td>Walking Bit (one)</td></tr>
<tr><td>1011b</td><td>Walking Zero</td></tr>
<tr><td>1100b</td><td>Reserved</td></tr>
<tr><td>1101b</td><td>Reserved</td></tr>
<tr><td>1110b</td><td>Reserved</td></tr>
<tr><td>1111b</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td></td><td>14:12</td><td><b>Pattern Horizontal Seed</b>
Pixel of the scan line to start on corresponding to DST X=0.</td></tr>
<tr><td></td><td>11</td><td><b>Tiling Enable</b><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>Tile-X or Tile-Y.</td></tr>
</table></td></tr>
<tr><td></td><td>10:8</td><td><b>Pattern Vertical Seed</b>
Scan line of the 8x8 pattern to start on corresponding to DST Y=0.</td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>05h</td><td></td></tr>
</table></td></tr>
<tr><td>1
BR13</td><td>31</td><td><b>Reserved</b><table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>30</td><td><b>Clipping Enabled</b><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><b>Reserved</b><table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>28</td><td><b>Mono Pattern Transparency Mode</b><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><b>Reserved</b></td></tr>
</table>

XY_MONO_PAT_FIXED_BLT

<table>
<tr><td></td><td></td><td><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>25:24</td><td><b>Color Depth</b><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><b>Raster Operation</b></td></tr>
<tr><td></td><td>15:0</td><td><b>Destination Pitch in DWords</b>
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><b>Destination Y1 Coordinate (Top)</b>
16 bit signed number.</td></tr>
<tr><td></td><td>15:0</td><td><b>Destination X1 Coordinate (Left)</b>
16 bit signed number.</td></tr>
<tr><td>3
BR23</td><td>31:16</td><td><b>Destination Y2 Coordinate (Bottom)</b>
16 bit signed number.</td></tr>
<tr><td></td><td>15:0</td><td><b>Destination X2 Coordinate (Right)</b>
16 bit signed number.</td></tr>
<tr><td>4
BR09</td><td>31:0</td><td><b>Destination Base Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr></table>Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes.</td></tr>
<tr><td>5
BR16</td><td>31:0</td><td><b>Pattern Background Color</b>
8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]</td></tr>
<tr><td>6
BR17</td><td>31:0</td><td><b>Pattern Foreground Color</b>
8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]</td></tr>
</table>

XY_MONO_SRC_COPY_BLT

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>0
BR00</td><td>31:29</td><td><b>Client</b>
<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></td><td>28:22</td><td><b>Instruction Target(Opcode)</b>
<table>
<tr><td>Default Value:</td><td>54h</td></tr>
<tr><td>Format:</td><td>Opcode</td></tr>
</table></td></tr>
<tr><td></td><td>21:20</td><td><b>32bpp Byte Mask</b>
This field is only used for 32bpp.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>00b</td><td><b>[Default]</b></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></td><td>19:17</td><td><b>Monochrome source data bit position of the first pixel within a byte per scan line.</b></td></tr>
<tr><td></td><td>16:12</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>11</td><td><b>Tiling Enable</b>
<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>Tile-X or Tile-Y.</td></tr>
</table></td></tr>
<tr><td></td><td>10:8</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b></td></tr>
</table>

XY_MONO_SRC_COPY_BLT

<table>
<tr><td></td><td></td><td><table><tr><th>Value</th><th>Name</th></tr><tr><td>06h</td><td></td></tr></table></td></tr>
<tr><td>1
BR13</td><td>31</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>30</td><td><b>Clipping Enabled</b><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><b>Mono Source Transparency Mode</b><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><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>25:24</td><td><b>Color Depth</b><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><b>Raster Operation</b></td></tr>
<tr><td></td><td>15:0</td><td><b>Destination Pitch in DWords</b>
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><b>Destination Y1 Coordinate (Top)</b>
16 bit signed number.</td></tr>
<tr><td></td><td>15:0</td><td><b>Destination X1 Coordinate (Left)</b>
16 bit signed number.</td></tr>
<tr><td>3
BR23</td><td>31:16</td><td><b>Destination Y2 Coordinate (Bottom)</b>
16 bit signed number.</td></tr>
<tr><td></td><td>15:0</td><td><b>Destination X2 Coordinate (Right)</b>
16 bit signed number.</td></tr>
<tr><td>4
BR09</td><td>31:0</td><td><b>Destination Base Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr></table>Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes.</td></tr>
<tr><td>5</td><td>31:0</td><td><b>Source Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr></table></td></tr>
</table>

XY_MONO_SRC_COPY_BLT

BR12		(address corresponding to DST X1,Y1) (Note no NPO2 change here).
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]

XY_MONO_SRC_COPY_IMMEDIATE_BLT

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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="5">0
BR00</td><td>31:29</td><td><b>Client</b>
<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><b>Instruction Target(Opcode)</b>
<table><tr><td>Default Value:</td><td>71h</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td>21:20</td><td><b>32bpp Byte Mask</b>
This field is only used for 32bpp.
<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td><b>[Default]</b></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><b>Monochrome source data bit position of the first pixel within a byte per scan line.</b></td></tr>
<tr><td>16:12</td><td><b>Reserved</b></td></tr>
</table>

<table>
<tr><th colspan="3">XY_MONO_SRC_COPY_IMMEDIATE_BLT</th></tr>
<tr><td></td><td></td><td><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11</td><td><b>Dest Tiling Enable</b><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>Tile-X or Tile-Y</td></tr></table></td></tr>
<tr><td></td><td>10:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>05h Excludes DWORD 0,1</td></tr></table>05 + DWL = (Number of Immediate double words)h</td></tr>
<tr><td>1
BR13</td><td>31</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>30</td><td><b>Clipping Enabled</b><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><b>Mono Source Transparency Mode</b><table><tr><th>Value</th><th>Name</th></tr><tr><td>0b</td><td>Transparency Enabled</td></tr><tr><td>1b</td><td>Use Background</td></tr></table></td></tr>
<tr><td></td><td>28:26</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>25:24</td><td><b>Color Depth</b><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><b>Raster Operation</b></td></tr>
<tr><td></td><td>15:0</td><td><b>Destination Pitch in DWords</b>
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><b>Destination Y1 Coordinate (Top)</b>
16 bit signed number.</td></tr>
<tr><td></td><td>15:0</td><td><b>Destination X1 Coordinate (Left)</b>
16 bit signed number.</td></tr>
<tr><td>3</td><td>31:16</td><td><b>Destination Y2 Coordinate (Bottom)</b>
16 bit signed number.</td></tr>
</table>

XY_MONO_SRC_COPY_IMMEDIATE_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. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes.
5 BR18	31:0	**Source Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
6 BR19	31:0	**Source Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
7..n	31:0	**Immediate Data**

XY_PAT_BLT_IMMEDIATE

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="7">0
BR00</td><td>31:29</td><td><b>Client</b>
<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><b>Instruction Target(Opcode)</b>
<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><b>32bpp Byte Mask</b>
This field is only used for 32bpp.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>00b</td><td><b>[Default]</b></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><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td>14:12</td><td><b>Pattern Horizontal Seed</b>
Pixel of the scan line to start on corresponding to DST X=0.</td></tr>
<tr><td>11</td><td><b>Tiling Enable</b>
<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>Tile-X or Tile-Y.</td></tr>
</table></td></tr>
<tr><td>10:8</td><td><b>Pattern Vertical Seed</b>
Scan line of the 8x8 pattern to start on corresponding to DST Y=0.</td></tr>
</table>

<table>
<tr><th colspan="3">XY_PAT_BLT_IMMEDIATE</th></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>03h Excludes DWORD 0,1</td></tr></table>03 + DWL = (Number of Immediate double)h</td></tr>
<tr><td rowspan="6">1
BR13</td><td>31</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>30</td><td><b>Clipping Enabled</b><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>29:26</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>25:24</td><td><b>Color Depth</b><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><b>Raster Operation</b></td></tr>
<tr><td>15:0</td><td><b>Destination Pitch in DWords</b>
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
BR22</td><td>31:16</td><td><b>Destination Y1 Coordinate (Top)</b>
16 bit signed number.</td></tr>
<tr><td>15:0</td><td><b>Destination X1 Coordinate (Left)</b>
16 bit signed number.</td></tr>
<tr><td rowspan="2">3
BR23</td><td>31:16</td><td><b>Destination Y2 Coordinate (Bottom)</b>
16 bit signed number.</td></tr>
<tr><td>15:0</td><td><b>Destination X2 Coordinate (Right)</b>
16 bit signed number.</td></tr>
<tr><td>4
BR09</td><td>31:0</td><td><b>Destination Base Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr></table>Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes.</td></tr>
<tr><td>5..n</td><td>31:0</td><td><b>Immediate Data</b></td></tr>
</table>

XY_PAT_BLT

Source:	BlitterCS
Length Bias:	2

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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0
BR00</td><td>31:29</td><td><b>Client</b>
<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></td><td>28:22</td><td><b>Instruction Target(Opcode)</b>
<table>
<tr><td>Default Value:</td><td>51h</td></tr>
<tr><td>Format:</td><td>Opcode</td></tr>
</table></td></tr>
<tr><td></td><td>21:20</td><td><b>32bpp Byte Mask</b>
This field is only used for 32bpp.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>00b</td><td><b>[Default]</b></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></td><td>19:15</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>14:12</td><td><b>Pattern Horizontal Seed</b>
Pixel of the scan line to start on corresponding to DST X=0.</td></tr>
<tr><td></td><td>11</td><td><b>Tiling Enable</b>
<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>Tile-X or Tile-Y.</td></tr>
</table></td></tr>
<tr><td></td><td>10:8</td><td><b>Pattern Vertical Seed</b>
Scan line of the 8x8 pattern to start on corresponding to DST Y=0.</td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b>
<table>
<tr><td>Default Value:</td><td>04h</td></tr>
</table></td></tr>
</table>

XY_PAT_BLT

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. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes.
5 BR15	31:0	**Pattern Base Address** Format: GraphicsAddress[31:0] (28:06 are implemented) (Note no NPO2 change here) . The pattern data must be located in linear memory.

XY_PAT_CHROMA_BLT

Source:	BlitterCS
Length Bias:	2

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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0
BR00</td><td>31:29</td><td><b>Client</b><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></td><td>28:22</td><td><b>Instruction Target(Opcode)</b><table><tr><td>Default Value:</td><td>76h</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td></td><td>21:20</td><td><b>32bpp Byte Mask</b>
This field is only used for 32bpp.<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td><b>[Default]</b></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></td><td>19:17</td><td><b>Transparency Range Mode</b>
(chroma-key) - Dst Chroma-key modes ONLY (SRC ILLEGAL)</td></tr>
<tr><td></td><td>16:15</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>14:12</td><td><b>Pattern Horizontal Seed</b>
Pixel of the scan line to start on corresponding to DST X=0.</td></tr>
<tr><td></td><td>11</td><td><b>Tiling Enable</b><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>Tile-X or Tile-Y.</td></tr></table></td></tr>
<tr><td></td><td>10:8</td><td><b>Pattern Vertical Seed</b>
Scan line of the 8x8 pattern to start on corresponding to DST Y=0.</td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>06h</td></tr></table></td></tr>
</table>

<table>
<tr><th colspan="3">XY_PAT_CHROMA_BLT</th></tr>
<tr><td rowspan="6">1
BR13</td><td>31</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>30</td><td><b>Clipping Enabled</b><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>29:26</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>25:24</td><td><b>Color Depth</b><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><b>Raster Operation</b></td></tr>
<tr><td>15:0</td><td><b>Destination Pitch in DWords</b>
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
BR22</td><td>31:16</td><td><b>Destination Y1 Coordinate (Top)</b>
16 bit signed number.</td></tr>
<tr><td>15:0</td><td><b>Destination X1 Coordinate (Left)</b>
16 bit signed number.</td></tr>
<tr><td rowspan="2">3
BR23</td><td>31:16</td><td><b>Destination Y2 Coordinate (Bottom)</b>
16 bit signed number.</td></tr>
<tr><td>15:0</td><td><b>Destination X2 Coordinate (Right)</b>
16 bit signed number.</td></tr>
<tr><td>4
BR09</td><td>31:0</td><td><b>Destination Base Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr></table>Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit 11 enabled), this address is limited to 4Kbytes.</td></tr>
<tr><td>5
BR15</td><td>31:0</td><td><b>Pattern Base Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr></table>(26:06 are used, other bits are ignored) (Note no NPO2 change here). The pattern data must be located in linear memory.</td></tr>
<tr><td>6
BR18</td><td>31:0</td><td><b>Transparency Color Low</b>
(Chroma-key Low = Pixel Greater or Equal)</td></tr>
</table>

XY_PAT_CHROMA_BLT		
7 BR19	31:0	**Transparency Color High** (Chroma-key High = Pixel Less or Equal)

XY_PAT_CHROMA_BLT_IMMEDIATE

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="7">0
BR00</td><td>31:29</td><td><b>Client</b>
<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><b>Instruction Target(Opcode)</b>
<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><b>32bpp Byte Mask</b>
This field is only used for 32bpp.
<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td><b>[Default]</b></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><b>Transparency Range Mode</b>
(chroma-key) - Dst Chroma-key modes ONLY (SRC ILLEGAL)</td></tr>
<tr><td>16:15</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>14:12</td><td><b>Pattern Horizontal Seed</b>
Pixel of the scan line to start on corresponding to DST X=0.</td></tr>
<tr><td>11</td><td><b>Tiling Enable</b>
<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>Tile-X or Tile-Y.</td></tr></table></td></tr>
</table>

<table>
<tr><th colspan="3">XY_PAT_CHROMA_BLT_IMMEDIATE</th></tr>
<tr><td rowspan="2"></td><td>10:8</td><td><b>Pattern Vertical Seed</b>
Scan line of the 8x8 pattern to start on corresponding to DST Y=0.</td></tr>
<tr><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>05h Excludes DWORD 0,1</td></tr></table>05 + DWL = (Number of Immediate double)h</td></tr>
<tr><td rowspan="6">1
BR13</td><td>31</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>30</td><td><b>Clipping Enabled</b><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>29:26</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>25:24</td><td><b>Color Depth</b><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><b>Raster Operation</b></td></tr>
<tr><td>15:0</td><td><b>Destination Pitch in DWords</b>
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><b>Destination Y1 Coordinate (Top)</b>
16 bit signed number.</td></tr>
<tr><td>15:0</td><td><b>Destination X1 Coordinate (Left)</b>
16 bit signed number.</td></tr>
<tr><td rowspan="2">3
BR23</td><td>31:16</td><td><b>Destination Y2 Coordinate (Bottom)</b>
16 bit signed number.</td></tr>
<tr><td>15:0</td><td><b>Destination X2 Coordinate (Right)</b>
16 bit signed number.</td></tr>
<tr><td>4
BR09</td><td>31:0</td><td><b>Destination Base Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr></table>Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes.</td></tr>
<tr><td>5
BR18</td><td>31:0</td><td><b>Transparency Color Low</b>
(Chroma-key Low = Pixel Greater or Equal)</td></tr>
</table>

XY_PAT_CHROMA_BLT_IMMEDIATE		
6 BR19	31:0	**Transparency Color High** (Chroma-key High = Pixel Less or Equal)
7..n	31:0	**Immediate Data**

XY_PIXEL_BLT

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 | 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

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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="7">0
BR00</td><td>31:29</td><td><b>Client</b><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><b>Instruction Target(Opcode)</b><table><tr><td>Default Value:</td><td>25h</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td>21:15</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>14:12</td><td><b>Pattern Horizontal Seed</b>
Pixel of the scan line to start on corresponding to DST X=0.</td></tr>
<tr><td>11</td><td><b>Tiling Enable</b><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>Tile-X or Tile-Y.</td></tr></table></td></tr>
<tr><td>10:8</td><td><b>Pattern Vertical Seed</b>
Scan line of the 8x8 pattern to start on corresponding to DST Y=0.</td></tr>
<tr><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>01h</td></tr></table></td></tr>
<tr><td rowspan="2">1
BR22</td><td>31:16</td><td><b>Destination Y1 Coordinate (Top)</b>
16 bit signed number.</td></tr>
<tr><td>15:0</td><td><b>Destination X1 Coordinate (Left)</b>
16 bit signed number.</td></tr>
<tr><td>2</td><td>31:16</td><td><b>Destination Y2 Coordinate (Bottom)</b>
16 bit signed number.</td></tr>
</table>

XY_SCANLINES_BLT		
BR23	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.

XY_SETUP_BLT

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).

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0
BR00</td><td>31:29</td><td><b>Client</b><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></td><td>28:22</td><td><b>Instruction Target(Opcode)</b><table><tr><td>Default Value:</td><td>01h</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td></td><td>21:20</td><td><b>32 bpp Byte Mask</b><table><tr><th>Value</th><th>Name</th></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></td><td>19:12</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11</td><td><b>Tiling Enable</b><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>Tile-X or Tile-Y.</td></tr></table></td></tr>
<tr><td></td><td>10:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>06h</td></tr></table></td></tr>
<tr><td>1
BR01</td><td>31</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>30</td><td><b>Clipping Enabled</b><table><tr><th>Value</th><th>Name</th></tr><tr><td>0b</td><td>Disabled</td></tr></table></td></tr>
</table>

<table>
<tr><th colspan="3">XY_SETUP_BLT</th></tr>
<tr><td></td><td></td><td><table><tr><td>1b</td><td>Enabled</td></tr></table></td></tr>
<tr><td></td><td>29</td><td><b>Mono Source Transparency Mode</b><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>28:26</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>25:24</td><td><b>Color Depth</b><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><b>Raster Operation</b></td></tr>
<tr><td></td><td>15:0</td><td><b>Destination Pitch in DWords</b>
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><b>ClipRect Y1 Coordinate (Top)</b>
(30:16 = 15 bit positive number)</td></tr>
<tr><td></td><td>15:0</td><td><b>ClipRect X1 Coordinate (Left)</b>
(14:00 = 15 bit positive number)</td></tr>
<tr><td>3
BR25</td><td>31:16</td><td><b>ClipRect Y2 Coordinate (Bottom)</b>
(30:16 = 15 bit positive number)</td></tr>
<tr><td></td><td>15:0</td><td><b>ClipRect X2 Coordinate (Right)</b>
(14:00 = 15 bit positive number)</td></tr>
<tr><td>4
BR09</td><td>31:0</td><td><b>Setup Destination Base Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr></table>Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes.</td></tr>
<tr><td>5
BR05</td><td>31:0</td><td><b>Setup Background Color</b>
8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0] All</td></tr>
<tr><td>6
BR06</td><td>31:0</td><td><b>Setup Foreground Color</b>
8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0] (SLB and TB only)</td></tr>
<tr><td>7
BR07</td><td>31:0</td><td><b>Setup Pattern Base Address for Color Pattern</b><table><tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr></table>(26:06 are implemented) (SLB only) (Note no NPO2 change here). The pattern data must be located in linear memory.</td></tr>
</table>

XY_SETUP_CLIP_BLT

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

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**		
			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:	07h
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
	29	**Reserved**		
			Format:	MBZ

XY_SETUP_MONO_PATTERN_SL_BLT

<table>
<tr><td></td><td>28</td><td><b>Mono Pattern Transparency Mode</b><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><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>25:24</td><td><b>Color Depth</b><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><b>Raster Operation</b></td></tr>
<tr><td></td><td>15:0</td><td><b>Destination Pitch in DWords</b>
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><b>ClipRect Y1 Coordinate (Top)</b>
(30:16 = 15 bit positive number)</td></tr>
<tr><td>15:0</td><td><b>ClipRect X1 Coordinate (Left)</b>
(14:00 = 15 bit positive number)</td></tr>
<tr><td rowspan="2">3
BR25</td><td>31:16</td><td><b>ClipRect Y2 Coordinate (Bottom)</b>
(30:16 = 15 bit positive number)</td></tr>
<tr><td>15:0</td><td><b>ClipRect X2 Coordinate (Right)</b>
(14:00 = 15 bit positive number)</td></tr>
<tr><td>4
BR09</td><td>31:0</td><td><b>Setup Destination Base Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr></table>Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes.</td></tr>
<tr><td>5
BR05</td><td>31:0</td><td><b>Setup Background Color</b>
8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0] All</td></tr>
<tr><td>6
BR06</td><td>31:0</td><td><b>Setup Foreground Color</b>
8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0] (SLB and TB only)</td></tr>
<tr><td>7
BR20</td><td>31:0</td><td><b>DW0 (least significant) for a Monochrome Pattern</b></td></tr>
<tr><td>8
BR21</td><td>31:0</td><td><b>DW1 (most significant) for a Monochrome Pattern</b></td></tr>
</table>

XY_SRC_COPY_BLT

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="6">0
BR00</td><td>31:29</td><td><b>Client</b><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><b>Instruction Target(Opcode)</b><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><b>32bpp Byte Mask</b>
This field is only used for 32bpp.<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td><b>[Default]</b></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><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>15</td><td><b>Src Tiling Enable</b><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>Tile-X or Tile-Y.</td></tr></table></td></tr>
<tr><td>14:12</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
</table>

XY_SETUP_MONO_PATTERN_SL_BLT

<table>
<tr><td></td><td>11</td><td><b>Dest Tiling Enable</b><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>Tile-X or Tile-Y.</td></tr></table></td></tr>
<tr><td></td><td>10:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><th>Value</th><th>Name</th></tr><tr><td>06h</td><td></td></tr></table></td></tr>
<tr><td>1
BR13</td><td>31</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>30</td><td><b>Clipping Enabled</b><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>29:26</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>25:24</td><td><b>Color Depth</b><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><b>Raster Operation</b></td></tr>
<tr><td></td><td>15:0</td><td><b>Destination Pitch in DWords</b>
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><b>Destination Y1 Coordinate (Top)</b>
16 bit signed number.</td></tr>
<tr><td></td><td>15:0</td><td><b>Destination X1 Coordinate (Left)</b>
16 bit signed number.</td></tr>
<tr><td>3
BR23</td><td>31:16</td><td><b>Destination Y2 Coordinate (Bottom)</b>
16 bit signed number.</td></tr>
<tr><td></td><td>15:0</td><td><b>Destination X2 Coordinate (Right)</b>
16 bit signed number.</td></tr>
<tr><td>4
BR09</td><td>31:0</td><td><b>Destination Base Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr></table>Base address of the destination surface: X=0, Y=0. When Dest Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes.</td></tr>
</table>

XY_SETUP_MONO_PATTERN_SL_BLT

5 BR26	31:16	**Source Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Source X1 Coordinate (Left)** 16 bit signed number.
6 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).
7 BR12	31:0	**Source Base Address** Format: GraphicsAddress[31:0] Base address of the destination surface: X=0, Y=0. When Src Tiling is enabled (Bit_15 enabled), this address is limited to 4Kbytes.

XY_SRC_COPY_CHROMA_BLT

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><b>Client</b>
<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><b>Instruction Target(Opcode)</b>
<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><b>32bpp Byte Mask</b>
This field is only used for 32bpp.
<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td><b>[Default]</b></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><b>Transparency Range Mode</b>
(chroma-key)</td></tr>
<tr><td>16</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>15</td><td><b>Src Tiling Enable</b>
<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>Tile-X or Tile-Y.</td></tr></table></td></tr>
<tr><td>14:12</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
</table>

<table>
<tr><th colspan="3">XY_SRC_COPY_CHROMA_BLT</th></tr>
<tr><td></td><td>11</td><td><b>Dest Tiling Enable</b>
<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>Tile-X or Tile-Y.</td></tr>
</table></td></tr>
<tr><td></td><td>10:8</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b>
<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><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>30</td><td><b>Clipping Enabled</b>
<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>29:26</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>25:24</td><td><b>Color Depth</b>
<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><b>Raster Operation</b></td></tr>
<tr><td></td><td>15:0</td><td><b>Destination Pitch in DWords</b>
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><b>Destination Y1 Coordinate (Top)</b>
16 bit signed number.</td></tr>
<tr><td></td><td>15:0</td><td><b>Destination X1 Coordinate (Left)</b>
16 bit signed number.</td></tr>
<tr><td>3
BR23</td><td>31:16</td><td><b>Destination Y2 Coordinate (Bottom)</b>
16 bit signed number.</td></tr>
<tr><td></td><td>15:0</td><td><b>Destination X2 Coordinate (Right)</b>
16 bit signed number.</td></tr>
<tr><td>4
BR09</td><td>31:0</td><td><b>Destination Base Address</b>
<table>
<tr><td>Format:</td><td>GraphicsAddress[31:0]</td></tr>
</table>
Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes.</td></tr>
</table>

XY_SRC_COPY_CHROMA_BLT

5 BR26	31:16	**Source Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Source X1 Coordinate (Left)** 16 bit signed number.
6 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).
7 BR12	31:0	**Source Base Address** Format: GraphicsAddress[31:0] Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_15 enabled), this address is limited to 4Kbytes.
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_TEXT_BLT

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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0
BR00</td><td>31:29</td><td><b>Client</b>
<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></td><td>28:22</td><td><b>Instruction Target(Opcode)</b>
<table>
<tr><td>Default Value:</td><td>26h</td></tr>
<tr><td>Format:</td><td>Opcode</td></tr>
</table></td></tr>
<tr><td></td><td>21:17</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>16</td><td><b>Bit / Byte Packed</b>
Byte packed is for the NT driver.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0</td><td>Bit</td></tr>
<tr><td>1</td><td>Byte</td></tr>
</table></td></tr>
<tr><td></td><td>15:12</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>11</td><td><b>Tiling Enable</b>
<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>Tile-X or Tile-Y.</td></tr>
</table></td></tr>
<tr><td></td><td>10:8</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b>
<table>
<tr><td>Default Value:</td><td>02h</td></tr>
</table></td></tr>
<tr><td>1</td><td>31:16</td><td><b>Destination Y1 Coordinate (Top)</b></td></tr>
</table>

XY_TEXT_BLT		
BR22		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 BR12	31:0	**Source Address** Format: GraphicsAddress[31:0] (address of the first byte on scan line corresponding to Dst X1,Y1). (Note no NPO2 change here)

XY_TEXT_IMMEDIATE_BLT

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** Format: MBZ
	16	**Bit / Byte Packed** Byte packed is for the NT driver. Value | Name 0 | Bit 1 | Byte
	15:12	**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

XY_TEXT_IMMEDIATE_BLT

	7:0	**DWord Length** Default Value: | 01h Excludes DWORD 0,1 01 + DWL = (Number of Immediate double words)h
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**