Intel® Iris® Plus Graphics and UHD Graphics Open Source

Programmer's Reference Manual

For the 2019 10th Generation Intel Core™ Processors based on the "Ice Lake" Platform

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

January 2020, Revision 1.0

Creative Commons License

You are free to Share - to copy, distribute, display, and perform the work under the following conditions:

- **Attribution.** You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
- **No Derivative Works.** You may not alter, transform, or build upon this work.

Notices and Disclaimers

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

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).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 3h 3DPRIMITIVE Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 0h 3DPRIMITIVE Format: OpCode
	15	**Reserved**
	14	**Reserved** Format: MBZ
	13	**Reserved** Format: MBZ
	12	**POSH Enable** POSH functionality is enabled for this 3DPRIMITIVE command when this bit is set. When executed from the POCS command stream, the 3DPRIMITIVE command is processed

and visibility information is recorded. When subsequently executed from the RCS command stream, the previously-recorded visibility information is used to optimize the processing for this 3DPRIMITIVE command.
Setting this bit will also (temporarily -- for the 3DPRIMITIVE command) inhibit modification of the following statistics counters when the 3DPRIMITIVE command is executed from the RCS command stream, even if these statistics counters are enabled via VF/VS state. This is done to prevent RCS from double-counting statistics that have already been counted by POCS.

- IA_VERTICES_COUNT
- IA_PRIMITIVES_COUNT
- VS_INVOCATION_COUNT

When this bit is clear, this 3DPRIMITIVE command is ignored by the POCS command stream, and when executed in the RCS command stream the command is processed without use of visibility information (as none will have been recorded by the POCS command stream) and with statistics counted as specified by VF/VS state.

Programming Notes

This bit must be clear (POSH disabled) when any one of the following conditions are met:

- A topology type other than: 3DPRIM_TRILIST, 3DPRIM_TRISTRIP, 3DPRIM_TRISTRIP_REVERSE
- UAV Coherency Required and/or Accesses UAV set in 3DSTATE_VS
- Tessellation enabled
- Geometry Shader enabled
- StreamOut enabled

For each 3DPRIMITIVE command that SW submits to the POCS command stream with POSH enabled, SW shall also submit an identical 3DPRIMITIVE command with POSH enabled to the RCS command stream, in the same sequence as submitted to the POCS command stream. In addition to the parameters passed in the 3DPRIMITIVE command fields, the following states shall also match:

- 3DSTATE_VF: States controlling cut index processing must match
- For RANDOM indexing, the Index Buffer states must match

11 **Extended Parameters Present**

Format:	Boolean

If true, three additional DWords containing XP0, XP1 and XP2 parameters are included in this command. Depending on the setting of Indirect Parameter Enable, XP0-2 parameters (sourced either from the inline XP0-2 DWords or indirectly from the 3DPRIM_XP0-2 registers) are passed to the VF stage as possible sources for VF SGV insertion.
If false, XP0-2 DWords are not included in this command and instead XP0-2 values default to zero if enabled as sources in VF SGV insertion.

3DPRIMITIVE

10	**Indirect Parameter Enable** Format: Enable **Description** 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 shall not be ENABLED at the same time, or behavior is UNDEFINED. If set and Extended Parameters Present is true, the current contents of the 3DPRIM_XP0-2 MMIO registers are passed to the VF stage as possible sources for VF SGV insertion and the Extended Parameter 0-2 values included in this command are ignored.
9	**UAV Coherency Required** Format: U1 SW will be required to set this bit if there is the possibility of sharing a UAV from a previous 3DPRIMITVE command. If set, this command may cause a flush due to UAV coherency requirements. If none of the shaders have UAV access enabled, then this bit is ignored.
8	**Predicate Enable** Format: Enable If set, this command is executed (or not) depending on the current value of the MI Predicate internal state bit. This command is ignored only if PredicateEnable is set and the Predicate state bit is 0.
7:0	**DWord Length** Format: =n Total Length - 2

Value	Name	Programming Notes	Exists If
5h	5		[Extended Parameter Enable] == 'false'
8h	8 **[Default]**	When Extended Parameter Enable is true, three Dwords containing Extended Parameter 0-2 shall	[Extended Parameter

				be included in this command and the DWord Length field set to account for these additional DWords.	Enable] == 'true'

1 — 31:10 — Reserved

Format:	MBZ

Description
Bits 31:28 are reserved for hardware use only.

1 — 9 — End Offset Enable

Format:	Enable

If set, the Vertex Count Per Instance field is IGNORED, and the 3DPRIM_END_OFFSET register is used to indirectly specify the vertex count by defining the amount of valid data in VB0. The following restrictions apply:

- VB0 must be enabled for use
- VertexAccessType = SEQUENTIAL
- Start Vertex Location = 0
- Start Instance Location = 0
- Base Vertex Location = 0

Vertices are output until EndOffset is reached or exceeded in VB0. If EndOffset is reached or exceeded within the data associated with a vertex, that vertex is considered incomplete and will not be output. Partial objects will be discarded (as is normally done).
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.

1 — 8 — Vertex Access Type

This field specifies how data held in vertex buffers marked as VERTEXDATA is accessed by Vertex Fetch.

Value	Name	Description
0h	SEQUENTIAL	VERTEXDATA buffers are accessed sequentiallyRequiref if End Offset Enable is ENABLED.
1h	RANDOM	VERTEXDATA buffers are accessed randomly via an index obtained from the Index Buffer.

1 — 7:6 — Reserved

Format:	MBZ

1 — 5:0 — Primitive Topology Type

Format:	3D_Prim_Topo_Type See table below for encoding, see 3D Overview for diagrams and general comments

3DPRIMITIVE

		Description This field specifies the topology type of 3D primitive generated by this command. Note that a single primitive topology (list/strip/fan/etc.) can contain a number of basic objects (lines, triangles, etc.). This field is ignored. The topology type is specified via the 3DSTATE_VF_TOPOLOGY command.
2	31:0	**Vertex Count Per Instance** Format: U32 Count of vertices This field specifies how many vertices are to be generated for each instance of the primitive topology. If End Offset Enable is clear: Format = U32 count of vertices Range = [0, 2^32-1] (upper limit probably constrained by VB size) Ignored if End Offset Enable or Indirect Parameter Enable is ENABLED. **Programming Notes** • This per-instance value should specify a valid number of vertices for the primitive topology type. E.g., for 3DPRIM_TRILIST_ADJ, this field should specify a multiple of 6 vertices. However, in cases where too few or too many vertices are provided, the unused vertices will be silently discarded by the pipeline. • A 0 value is this field effectively makes the command a 'no-operation'.
3	31:0	**Start Vertex Location** Format: U32 structure index This field specifies the "starting vertex" for each instance. This allows skipping over part of the vertices in a buffer if, for example, a previous 3DPRIMITIVE command had already drawn the primitives associated with the earlier entries. For SEQUENTIAL access, this field specifies, for each instance, a starting structure index into the vertex buffers For RANDOM access, this field specifies, for each instance, a starting index into the Index Buffer. **Programming Notes** • 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
4	31:0	**Instance Count** Format: U32 Count of instances 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

3DPRIMITIVE

ENABLED.

Value	Name
[0, FFFFFFFFh]	

5	31:0	**Start Instance Location** Format: U32 structure index **Description** This field specifies the "starting instance" for the command as an initial structure index into Vertex Buffers for vertex elements with InstancingEnable set. Subsequent instances will access sequential instance data structures, as controlled by the Instance Data Step Rate. **Programming Notes** • 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.
6	31:0	**Base Vertex Location** Format: S31 index structure bias 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. **Programming Notes** • 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.
7 **Exists if:** [Extended Parameters Present] == TRUE	31:0	**Extended Parameter 0** Format: U32 If Indirect Parameter Enable is not set, this field specifies a U32 XP0 parameter value to be passed to the VF stage as a possible source for SGV insertion. If Indirect Parameter Enable is set, this field is ignored.
8	31:0	**Extended Parameter 1**

3DPRIMITIVE		
Exists if: [Extended Parameters Present] == TRUE		Format: U32 If Indirect Parameter Enable is not set, this field specifies a U32 XP1 parameter value to be passed to the VF stage as a possible source for SGV insertion. If Indirect Parameter Enable is set, this field is ignored.
9 **Exists if:** [Extended Parameters Present] == TRUE	31:0	**Extended Parameter 2** Format: U32 If Indirect Parameter Enable is not set, this field specifies a U32 XP2 parameter value to be passed to the VF stage as a possible source for SGV insertion. If Indirect Parameter Enable is set, this field is ignored.

3DSTATE_3D_MODE

3DSTATE_3D_MODE

Source: RenderCS

Length Bias: 2

This command is general 3D programming state that can be shared from the top to bottom of the pipeline.

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 GFXPIPE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 1Eh 3DSTATE_3D_MODE Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:16	**Mask Bits 1** Format: Enable[16] This field is the mask bits for the state bits below. This is a bit wise mask where the bit number-16 is the value of the corresponding bit being masked in the same data word. For example, if you want to update state for bits 3:2 then bits 19:18 must be set.
	15:11	**Reserved** Format: MBZ
	10	**Reserved** Format: MBZ
	9	**Reserved**
	8	**Reserved**

3DSTATE_3D_MODE

7 Reserved

Format:	MBZ

6 Slice Hashing Table Enable

Format:	Enable

This field enables the use of indirect state **SLICE_HASH_TABLE** programmed via **3DSTATE_SLICE_HASH_STATE_POINTER**.

Value	Name	Description
0h	Disable **[Default]**	Slice Hashing is based on default lookup tables with the following function (X+Y) % total_enabled_subslices % enabled_slices
1h	Enable	Slice Hashing is via SLICE_HASH_TABLE from 3DSTATE_SLICE_HASH_STATE_POINTER[total_enabled_subslices-4]

5 Subslice Hashing Table Enable

Format:	Enable

This field enables the use of the Subslice Hashing Mode table programmed via **3DSTATE_SUBSLICE_HASH_TABLE**.

Value	Name	Description
0h	Disable **[Default]**	Subslice Hashing is computed
1h	Enable	Subslice Hashing is via 3DSTATE_SUBSLICE_HASH_TABLE

4 3D Scoreboard Hashing Mode

Value	Name	Description
0	Scoreboard address calculation optimized for Subslice Hashing Mode **[Default]**	Before scoreboard address calculations one address bit will be removed to increase the efficiency of the scoreboard.
1	Scoreboard address calculation not optimized	All address bits used when calculating scoreboard address.

Programming Notes
When **Subslice Hashing Table Enable** and table entries do not contain a checkerboard pattern for each subslice, then optimizing for **Subslice Hashing Mode** may result in reduced performance due to increased false dependencies.

3:2 Subslice Hashing Mode

This field defines hashing modes across subslices.

Value	Name	Programming Notes
0	8x8 hashing **[Default]**	
1	16X4 hashing	Hardware treats as code 0 (8x8 hashing) for backwards compatibility.
2	8X4 hashing	Hardware treads as code 0 (8x8 hashing) for backwards compatibility.

		3	16x16 hashing	

Programming Notes
A stalling flush is required before changing the value of this field. This is to make sure the entire pipeline is drained.

1:0 **Cross Slice Hashing Mode**

Format:	U2

Value	Name	Description	Programming Notes
0h	Normal Mode **[Default]**	num_slices > 1 : 16x16 Hashing enabled num_slices == 1: No cross slice hashing	
1h	Cross Slice Hashing Disable	Disables the cross slice hashing	
2h	Reserved	Reserved	
3h	32X32 hashing	32X32 pixel hashing across slices	This setting must be used when sub-slice hashing mode is 16x16 and num_slices > 1

3DSTATE_AA_LINE_PARAMETERS

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.

Workaround
Workaround: This command must be followed by a PIPE_CONTROL with CS Stall bit set.,

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 0Ah 3DSTATE_AA_LINE_PARAMETERS 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	**AA Point Coverage Bias** Format: U0.8 This field specifies the bias term to be used in the aa coverage computation for edges 0 and 3.
	23:16	**AA Coverage Bias** Format: U0.8 This field specifies the bias term to be used in the aa coverage computation for edges 0 and 3.
	15:8	**AA Point Coverage Slope**

3DSTATE_AA_LINE_PARAMETERS		
		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 coverage computation (though still output expanded U0.8 coverage values).
	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 coverage computation (though still output expanded U0.8 coverage values).
2	31:24	**AA Point 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.
	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	**AA Point 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.
	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

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	31:0	**Binding Table Pointers State Body** Format: 3DSTATE_BINDING_TABLE_POINTERS_BODY

3DSTATE_BINDING_TABLE_POINTERS_GS

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	31:0	**Binding Table Pointers State Body** Format: 3DSTATE_BINDING_TABLE_POINTERS_BODY

3DSTATE_BINDING_TABLE_POINTERS_HS

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	31:0	**Binding Table Pointers State Body** Format: 3DSTATE_BINDING_TABLE_POINTERS_BODY

3DSTATE_BINDING_TABLE_POINTERS_PS

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	**Reserved**
	14:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:0	**Binding Table Pointers State Body** Format: 3DSTATE_BINDING_TABLE_POINTERS_BODY

3DSTATE_BINDING_TABLE_POINTERS_VS

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	31:0	**Binding Table Pointers State Body** Format: 3DSTATE_BINDING_TABLE_POINTERS_BODY

3DSTATE_BINDING_TABLE_POOL_ALLOC

3DSTATE_BINDING_TABLE_POOL_ALLOC

Source: RenderCS

Length Bias: 2

Description

This command is to program the base address and size of the binding table pool. The address to fetch the binding table is based on the Binding Table Pool Base Address and the binding table pointer if the Binding Table Pool is enabled. Otherwise the binding table pointer is an offset from the Surface Base Address.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 19h 3DSTATE_BINDING_TABLE_POOL_ALLOC Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Value / Name 2h DWORD_COUNT_n **[Default]**
1..2	63:12	**Binding Table Pool Base Address** Format: GraphicsAddress63-12 Specifies the 4K-byte aligned base address for Binding Table Pool. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].
	11	**Binding Table Pool Enable** Format: U1 When this bit is set it enables HW generation of binding tables. When this bit is cleared it

disables HW generation of binding tables.

Programming Notes

If the Binding Table Pool Enable is set when resource streamer is disabled, the engine will fetch binding table entries from the Binding Table Pool Address. In this case, driver must ensure the pool is populated and the binding table is coherent prior to engine fetching binding table entries. The format of the binding table must match the same format output by the resource streamer hardware.

	10	**Reserved**
	9:7	**Reserved**
	6:0	**Surface Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this surface. **Programming Notes** Bit 0 is not programmable and is always zero.
3	31:12	**Binding Table Pool Buffer Size** Format: U20 This field specifies the size of the buffer in 4K pages. Any access which straddle or go past the end of the buffer will return 0.

Value	Name	Description
[0,1048575]		
0	No Valid Data	There is no valid data in the buffer

Restriction

Programming size of zero is illegal in the case that the pool is enabled.

	11:0	**Reserved** Format: MBZ

3DSTATE_BLEND_STATE_POINTERS

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.

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	31:0	**Blend State Pointer State Body** Format: 3DSTATE_BLEND_STATE_POINTERS_BODY

3DSTATE_CC_STATE_POINTERS

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.		
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	31:0	**CC State Pointers Body** Format: 3DSTATE_CC_STATE_POINTERS_BODY

3DSTATE_CHROMA_KEY

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, ValidValues : 0,1,2,3 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 considered "matching the key" if each component of the texel falls within the (inclusive) chroma range.

Programming Notes

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

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

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

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

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

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

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

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

3DSTATE_CLEAR_PARAMS

3DSTATE_CLEAR_PARAMS

Source: RenderCS

Length Bias: 2

Description

This command defines the depth clear value delivered as a pipelined state command. However, the state change pipelining isn't completely transparent (see restriction below).

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

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..2	63:0	**Clear Params State Body** Format: 3DSTATE_CLEAR_PARAMS_BODY

3DSTATE_CLIP

3DSTATE_CLIP		
Source:	RenderCS	
Length Bias:	2	
Restriction		
Restriction : When executed in POCS command stream, this programs the state for CLR stage of the POCS pipeline		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 12h 3DSTATE_CLIP Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 02h Excludes DWord (0,1) Format: =n Total Length - 2
1..3	95:0	**Clip State Body** Format: 3DSTATE_CLIP_BODY

3DSTATE_CONSTANT_DS

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).

Workaround

Workaround:
The driver must ensure The following case does not occur without a flush to the 3D engine: 3DSTATE_CONSTANT_* with buffer 3 read length equal to zero committed followed by a 3DSTATE_CONSTANT_* with buffer 0 read length not equal to zero committed. Possible ways to avoid this condition include:

- always force buffer 3 to have a non zero read length
- always force buffer 0 to a zero 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	**Reserved** Format: MBZ
	14:8	**Constant Buffer Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for all constant buffers defined in this command.
	7:0	**DWord Length** Format: =n Total Length - 2

3DSTATE_CONSTANT_DS

		Value / **Name** 9h — Excludes DWord (0,1) **[Default]**
1..10	319:0	**Constant Body** Format: **3DSTATE_CONSTANT(Body)** See the 3DSTATE_CONSTANT(Body) format for the shared portion of the 3DSTATE_CONSTANT command for VS, HS, DS, and GS.

3DSTATE_CONSTANT_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).

Workaround

Workaround:
The driver must ensure The following case does not occur without a flush to the 3D engine: 3DSTATE_CONSTANT_* with buffer 3 read length equal to zero committed followed by a 3DSTATE_CONSTANT_* with buffer 0 read length not equal to zero committed. Possible ways to avoid this condition include:

- always force buffer 3 to have a non zero read length
- always force buffer 0 to a zero 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	**Constant Buffer Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for all constant buffers defined in this command.
	7:0	**DWord Length** Format: =n Total Length - 2

3DSTATE_CONSTANT_GS

		Value / **Name** 9h — Excludes DWord (0,1) **[Default]**
1..10	319:0	**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

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).

Workaround

Workaround:
The driver must ensure The following case does not occur without a flush to the 3D engine: 3DSTATE_CONSTANT_* with buffer 3 read length equal to zero committed followed by a 3DSTATE_CONSTANT_* with buffer 0 read length not equal to zero committed. Possible ways to avoid this condition include:

- always force buffer 3 to have a non zero read length
- always force buffer 0 to a zero 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	**Reserved** Format: MBZ
	14:8	**Constant Buffer Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for all constant buffers defined in this command.
	7:0	**DWord Length** Format: =n Total Length - 2

3DSTATE_CONSTANT_HS

		<table><tr><th>Value</th><th>Name</th></tr><tr><td>9h</td><td>Excludes DWord (0,1) **[Default]**</td></tr></table>
1..10	319:0	**Constant Body** <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

3DSTATE_CONSTANT_PS

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

A 3DSTATE_GATHER_PS command must be dispatched along with any 3DSTATE_CONSTANT_PS command when the Gather Pool is enabled within a batch buffer.

The 3DSTATE_CONSTANT_* command is not committed to the shader unit until the corresponding (same shader) 3DSTATE_BINDING_TABLE_POINTER_* command is parsed. For example, the 3DSTATE_CONSTANT_VS command will not fetch the constant buffers from memory and make available to the shader until the 3DSTATE_BINDING_TABLE_POINTERS_VS is parsed by the render command streamer. In case of multiple 3DSTATE_CONSTANT_VS programmed prior to 3DSTATE_BINDING_TABLE_POINTER_VS, only the most recently programmed 3DSTATE_CONSTANT_VS will be committed.
On usage model of enabling legacy mode is when Resource Streamer is not enabled.

Workaround

Workaround:
The driver must ensure The following case does not occur without a flush to the 3D engine: 3DSTATE_CONSTANT_* with buffer 3 read length equal to zero committed followed by a 3DSTATE_CONSTANT_* with buffer 0 read length not equal to zero committed. Possible ways to avoid this condition include:

- always force buffer 3 to have a non zero read length
- always force buffer 0 to a zero 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

3DSTATE_CONSTANT_PS

		Format: OpCode
	15	**Reserved** Format: MBZ
	14:8	**Constant Buffer Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for all constant buffers defined in this command.
	7:0	**Dword Length** Format: =n Total Length - 2 Value: 9h — Name: Excludes DWord (0,1) **[Default]**
1..10	319:0	**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

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

When executed in the POCS command stream, this command programs state for the VSR stage of the POCS pipeline.

Workaround

Workaround:
The driver must ensure The following case does not occur without a flush to the 3D engine: 3DSTATE_CONSTANT_* with buffer 3 read length equal to zero committed followed by a 3DSTATE_CONSTANT_* with buffer 0 read length not equal to zero committed. Possible ways to avoid this condition include:

- always force buffer 3 to have a non zero read length
- always force buffer 0 to a zero 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: 15h 3DSTATE_CONSTANT_VS Format: OpCode
	15	**Reserved** Format: MBZ
	14:8	**Constant Buffer Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE

3DSTATE_CONSTANT_VS

		Specifies the memory object control state for all constant buffers defined in this command.
	7:0	**DWord Length** Format: =n Total Length - 2 **Value** / **Name** 9h — Excludes DWord (0,1) **[Default]**
1..10	319:0	**Constant Body** Format: **3DSTATE_CONSTANT(Body)** Following table is the shared portion of the 3DSTATE_CONSTANT command for VS, HS, DS, and GS

3DSTATE_CPS

3DSTATE_CPS		
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: 22h 3DSTATE_CPS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 7h Excludes DWord (0,1) Format: =n Total Length - 2
1..8	255:0	**CPS State Body** Format: 3DSTATE_CPS_BODY

3DSTATE_DEPTH_BUFFER

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).

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 5h 3DSTATE_DEPTH_BUFFER Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Excludes DWord(0,1) Value: 6h — Name: Excludes Dword (0,1) **[Default]**
1	31:29	**Surface Type** Value: 0h — Name: Reserved — Description: Reserved Value: 1h — Name: SURFTYPE_2D — Description: Defines a 2-dimensional map or array of maps Value: 2h — Name: Reserved — Description: Reserved Value: 3h — Name: SURFTYPE_CUBE — Description: Defines a cube map Value: 4h-6h — Name: Reserved — Description:

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.
1D surface type not allowed for depth surface and stencil surface.

Workaround
Workaround : If depth/stencil is enabled with 1D render target, depth/stencil surface type needs to be set to 2D surface type and height set to 1. Depth will use (legacy) TileY and stencil will use TileW. For this case only, the Surface Type of the depth buffer can be 2D while the Surface Type of the render target(s) are 1D, representing an exception to a programming note above.

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.

26:24 **Reserved**

Format:	MBZ

23 **Corner Texel Mode**

Format:	Enable

This field, when ENABLED, indicates when a surface is using corner texel-mode for depth surface. This bit changes how the size of each MIP when calculating the offset within a surface.

Value	Name	Description
0h	Disable **[Default]**	Corner Texel mode is not enabled.
1h	Enable	Corner Texel Mode is enabled.

22 **Hierarchical Depth Buffer Enable**

Format:	Enable

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

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

3DSTATE_DEPTH_BUFFER

This field must be disabled if surface type field is SURFTYPE_1D

21 **Reserved**

Format:	MBZ

20:18 **Surface Format**

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

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

17:0 **Surface Pitch**

Format:	U18-1 Pitch in (Bytes-1)

Description

For TileYF and TileYS surfaces, the range is dependent on the Cu parameter (refer to *Memory Data Formats* section for the definition of the Cu parameter depending on the case). The range in bytes is [2^{Cu}-1,262143] -> [(2^{Cu})B,256KB] = [1 tile, 256KB/(2^{Cu}) tiles]

Value	Name
[7Fh,3FFFFh]	

Programming Notes

The pitch specified must be a multiple of the tile pitch, in the range [128B, 128KB].
The minimum pitch should be calculated as per the formula given below.

pixel_size	Depth Format
2	Z16
4	Z24
4	Z32

The minimum pitch should be calculated based on Cu, Cv, W_L.
The Cu, Cv are the tile constants and $W_L$is the aligned width adjusted for MSAA.
Then use this for pitch formula :
Minimum_pitch =(ceiling((W_0 pixel_size) / (1 « Cu)) *(1 « Cu))– 1 ; //$W_0$is the aligned width for the largest LOD (i.e LOD 0)*
(1 « Cu) = tile width in bytes
(1 « Cv) = tile height in lines

3DSTATE_DEPTH_BUFFER

2..3	63:0	**Surface Base Address** This field specifies the address of the buffer in mapped Graphics Memory Format: **GraphicsAddress63-0** **Programming Notes** The Depth Buffer can only be mapped to Main Memory (uncached). If the surface is tiled, the base address must conform to the Per-Surface Tiling Alignment Rules. If the buffer is linear, the surface must be 64-byte aligned.
4	31:18	**Height** Format: U14-1 This field specifies the height of the surface. If the surface is MIP-mapped, this field contains the height of the base MIP level. **Programming Notes** The Height of the depth buffer must be the same as the Height of the render target(s) (defined in SURFACE_STATE), unless Surface Type is SURFTYPE_1D or SURFTYPE_2D with Depth = 0 (non-array) and LOD = 0 (non-mip mapped).
	17:4	**Width** Format: U14-1 This field specifies the width of the surface. If the surface is MIP-mapped, this field specifies the width of the base MIP level. The width is specified in units of pixels.

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

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

[0,16383]	Legal Range	Width of surface - 1 (x/u dimension)	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_CUBE')

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

3:0 **LOD**

Format:	U4 for LOD units

Value	Name
[0,14]	

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

5 31:21 **Depth**

Format:	U11-1

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

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

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

20:10 **Minimum Array Element**

Format:	U11

For 1D and 2D Surfaces:
This field indicates the minimum array element that can be accessed as part of this surface. The

delivered array index is added to this field before being used to address the surface.

For 3D Surfaces

This field indicates the minimum 'R' coordinate on the LOD currently being rendered to. This field is added to the delivered array index before it is used to address the surface.

For Other Surfaces

This field is ignored

Value	Name	Exists If
[0,2047]	SURFTYPE_1D/2D	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_1D'|Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_2D')
[0,2047]	SURFTYPE_3D	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_3D')

9:7 **Reserved**

Format:	MBZ

6:0 **Depth Buffer Object Control State**

Format:	MEMORY_OBJECT_CONTROL_STATE

Specifies the memory object control state for the depth buffer.

6 31:30 **Tiled Resource Mode**

For Depth Buffer Surfaces: This field specifies the tiled resource mode. **For other surfaces:** This field is ignored.

Value	Name	Description
0h	NONE	No tiled resource
1h	TILEYF	4KB tiled resources
2h	TILEYS	64KB tiled resources
3h	Reserved	

Programming Notes
If **Tile Mode** is not set to TILEMODE_YMAJOR, this field must be set to TRMODE_NONE.
HIZ and Stencil surfaces can't be tiled resource, hence will continue to follow legacy tiledY and tiledW respectively.

29:26 **Mip Tail Start LOD**

Format:	U4 in LOD Units

For Sampling Engine, Render Target, and Typed Surfaces: This field indicates which LOD is the first one in the MIP tail if **Tiled Mode** is not TRMODE_NONE. The MIP tail has a different layout than the rest of the surface. Refer to the *Memory Data Formats* section for more details. **For other surfaces:** This field is ignored.

Programming Notes
This field must be zero if the **Surface Format** is MONO8

This field is ignored if **Tiled Mode** is TRMODE_NONE unless **Surface Type** is SURFTYPE_1D.

If **Tiled Mode** is not TRMODE_NONE, this field must be set to ensure that mips within the mip tail do not overlap given the storage algorithms given in the Memory Data Formats section. The following table indicates the *maximum* size of the mip that is set to be the Mip Tail Start LOD for various cases:

Tiling Mode	Slot Size in Bytes	8-bit Size	16-bit Size	32-bit Size
2D TIleYs 1x	32KB	(128, 256)	(128, 128)	(64, 128)
2D TileYf 1x	2KB	(32, 64)	(32, 32)	(16, 32)

25:0 **Reserved**

Format:	MBZ

7 31:21 **Render Target View Extent**

Format:	U11-1

Value	Name	Description	Exists If
[0,2047]	Legal Range	Number of array elements- 1	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_1D')
[0,2047]	Legal Range	Number of array elements- 1	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_2D')
[0,2047]	Legal Range	To indication extent of [1,2048]	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_3D')
[0,0]	Legal Range	Must be zero	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_CUBE')

20:15 **Reserved**

14:0 **Surface QPitch**

Format:	U15[16:2]

Description

The interpretation of this field is dependent on **Surface Type** as follows:

- SURFTYPE_1D: distance in *pixels* between array slices
- SURFTYPE_2D/CUBE: distance in *rows* between array slices
- SURFTYPE_3D: distance in *rows* between R-slices

For 1D and 2D Surfaces:
This field must be set to the same value as the Depth field.
For 3D Surfaces:
This field indicates the extent of the accessible 'R' coordinates minus 1 on the LOD currently being rendered to.
For Other Surfaces
This field is ignored.

Format:
`QPitch[16:2]`

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

Programming Notes

Software must ensure that this field is set to a value sufficiently large that array slices in the surface do not overlap. Refer to the *Memory Data Formats* section for information on how surfaces are stored.
TYS/TYF QPitch is valid only for 2D array surfaces and represents the tile-padded total number of texels(lines) in a single array slice.
Height of each LOD:
HL = AlignToTileHeight(MSAA_height_factor * (**height** »L) > 0? **height**»L : 1), where AlignToTileHeight(x) is (ceiling((x) / (1 « Cv)) *(1 « Cv))
Height of all LODs is a sum:
H = H0+H1+..Hn,
N is number of mip levels.
If surface has MIP tail, equation stops at Hn where n=MipTailStartLOD. MipTail is single tile.
QPitch is multiple of tile height (1 « Cv) and should be equal or greater H computed above.

3DSTATE_DRAWING_RECTANGLE

3DSTATE_DRAWING_RECTANGLE

Source: RenderCS

Length Bias: 2

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

Restriction

Restriction : When executed in POCS command stream, this programs the drawing rectangle for the SFR stage of the POCS pipeline.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 00h 3DSTATE_DRAWING_RECTANGLE Format: OpCode
	15:14	**Core Mode Select** Format: U2 Specifies which core this command will be considered valid and update based on the state in this command. Value / Name / Description 0h / Legacy / Both cores are enabled and will update the state. 1h / Core 0 Enabled / State will be updated in Core 0 only 2h / Core 1 Enabled / State will be updated in Core 1 only 3h / Reserved /
	13:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 2h Excludes DWord (0,1) Format: =n Total Length - 2

3DSTATE_DRAWING_RECTANGLE

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.

1 — 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
[0,16383]	Device ignores bits 15:14

Programming Notes

This value can be larger than Clipped Drawing Rectangle X Max. If Xmin>Xmax, the clipped drawing rectangle is null, all polygons are discarded. If Xmin==Xmax, the clipped drawing rectangle is 1 pixel wide in the X direction.

2 — 31:16 — Clipped Drawing Rectangle Y Max

Format:	U16 in Pixels from Color Buffer origin (upper left corner)

Specifies Ymax value of (inclusive) intersection of Drawing rectangle with the Color (Destination) Buffer, used for clipping. Pixels with coordinates greater than Ymax will be clipped out.

Value	Name
[0,16383]	Device ignores bits 31:30

Programming Notes

This value can be less than Clipped Drawing Rectangle Y Min. If Ymax<Ymin, the clipped drawing rectangle is null, all polygons are discarded. If Ymin==Ymax, the clipped drawing rectangle is 1 pixel wide in the Y direction.

2 — 15:0 — Clipped Drawing Rectangle X Max

Format:	U16 in Pixels from Color Buffer origin (upper left corner)

Specifies Xmax value of (inclusive) intersection of Drawing rectangle with the Color (Destination) Buffer, used for clipping. Pixels with coordinates greater than Xmax will be clipped out.

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

3DSTATE_DRAWING_RECTANGLE

		Programming Notes 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.
3	31:16	**Drawing Rectangle Origin Y** Format: S15 in Pixels from Color Buffer origin (upper left corner). **Description** Range: [-16384,16383] (Bit 31 should be a sign extension) Specifies Y origin of Drawing Rectangle (in whole pixels) relative to origin of the Color Buffer, used to map incoming (Draw Rectangle-relative) vertex positions to the Color Buffer space.
	15:0	**Drawing Rectangle Origin X** Format: S15 in Pixels from Color Buffer origin (upper left corner). **Description** Range: [-16384,16383] (Bit 15 should be a sign extension) Specifies X origin of Drawing Rectangle (in whole pixels) relative to origin of the Color Buffer, used to map incoming (Draw Rectangle-relative) vertex positions to the Color Buffer space.

3DSTATE_DS

3DSTATE_DS

Source: RenderCS

Length Bias: 2

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 1Dh 3DSTATE_DS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 9h Excludes DWord (0,1) Format: =n Total Length - 2
1..10	319:0	**DS State Body** Format: 3DSTATE_DS_BODY

3DSTATE_GS

3DSTATE_GS		
Source: RenderCS Length Bias: 2		
Controls the GS stage hardware.		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 11h 3DSTATE_GS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 8h Excludes DWord (0,1) Format: =n
1..9	287:0	**GS State Body** Format: 3DSTATE_GS_BODY

3DSTATE_HIER_DEPTH_BUFFER

3DSTATE_HIER_DEPTH_BUFFER

Source: RenderCS

Length Bias: 2

Description

This command sets the surface state of the hierarchical depth buffer, delivered as a pipelined state command. However, the state change pipelining isn't completely transparent (see restriction below).

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

Programming Notes

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 07h 3DSTATE_HIER_DEPTH_BUFFER Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**Dword Length** Format: =n Total Length - 2 Value / Name

3DSTATE_HIER_DEPTH_BUFFER

		3h: Excludes Dword (0,1) **[Default]**
1	31:25	**Hierarchical Depth Buffer Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for the hierarchical depth buffer.
	24:23	**Reserved** Format: MBZ
	22	**Corner Texel Mode** Format: Enable This field, when ENABLED, indicates when a surface is using corner texel-mode for depth and HIZ surface. This bit changes how the size of each MIP when calculating the offset within a surface. **Value** / **Name** / **Description** 0h / Disable **[Default]** / Corner texel mode is disabled by default. 1h / Enable / Corner Texel mode is enabled.
	21:17	**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, 1FFFFh] / 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..3	63:0	**Surface Base Address** Format: GraphicsAddress[63:0]HierarchicalDepthBuffer This field specifies the address of the buffer in Graphics Memory. **Programming Notes** The Hierarchical Depth Buffer can only be mapped to Main Memory (uncached).
4	31:15	**Reserved** Format: MBZ
	14:0	**Surface QPitch** Format: U15[16:2] **Description**

The interpretation of this field is dependent on **Surface Type** as follows:

- SURFTYPE_1D: distance in *pixels* between array slices
- SURFTYPE_2D/CUBE: distance in *rows* between array slices
- SURFTYPE_3D: distance in *rows* between R-slices

Format:

```
QPitch[16:2]
```

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

Programming Notes

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

3DSTATE_HS

3DSTATE_HS		
Source: RenderCS Length Bias: 2		
Controls the HS stage hardware.		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 1Bh 3DSTATE_HS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Value: 7 — Name: Excludes DWord (0,1) **[Default]**
1..8	255:0	**HS State Body** Format: 3DSTATE_HS_BODY

3DSTATE_INDEX_BUFFER

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.

Programming Notes

When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 0Ah 3DSTATE_INDEX_BUFFER Format: OpCode
	15	**Reserved**
	14:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 3h Excludes DWord (0,1) Format: =n Total Length - 2
1..4	127:0	**Index Buffer State Body** Format: 3DSTATE_INDEX_BUFFER_BODY

3DSTATE_LINE_STIPPLE

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.

Workaround

Workaround: This command must be followed by a PIPE_CONTROL with CS Stall bit set.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 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** It is provided only for HW-generated commands as part of context save/restore. SW must initialize the current repeat counter, current stipple count fields if it sets this bit to enable. SW must set this bit to reset the stipple count.
	30	**Reserved** Format: MBZ

3DSTATE_LINE_STIPPLE

	29:21	**Current Repeat Counter** Format: U9 This field sets the HW-internal repeat counter state. SW must initialize it to 1 if the modify enable is set.
	20	**Reserved** Format: MBZ
	19:16	**Current Stipple Index** Format: U4 This field sets the HW-internal stipple pattern index. SW must initialize it to 0 if the modify enable is set.
	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: [1, 256] Name:

3DSTATE_MONOFILTER_SIZE

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.

Workaround

Workaround: This command must be followed by a PIPE_CONTROL with CS Stall bit set.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 11h 3DSTATE_MONOFILTER_SIZE Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:6	**Reserved** Format: MBZ
	5:3	**Monochrome Filter Width** Format: U3 This field specifies the width of the monochrome filter. It is ignored if the monochrome filter is not enabled. Value: [1,7] — Name:

3DSTATE_MONOFILTER_SIZE		
	2:0	**Monochrome Filter Height** Format: U3 This field specifies the height of the monochrome filter. It is ignored if the monochrome filter is not enabled. **Value** / **Name** [1,7]

3DSTATE_MULTISAMPLE

3DSTATE_MULTISAMPLE

Source: RenderCS

Length Bias: 2

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

Programming Notes

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

Restriction : When executed in the POCS command stream, this command programs the multisample state for the Raster stage of the POCS pipeline

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 0Dh 3DSTATE_MULTISAMPLE Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:0	**Multisample State Body** Format: 3DSTATE_MULTISAMPLE_BODY

3DSTATE_POLY_STIPPLE_OFFSET

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.

Workaround
Workaround: This command must be followed by a PIPE_CONTROL with CS Stall bit set.,

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 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

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

3DSTATE_POLY_STIPPLE_PATTERN

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..32	1023:0	**Pattern Row** Format: U32[32] 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_BLEND

3DSTATE_PS_BLEND		
Source: RenderCS Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 4Dh 3DSTATE_PS_BLEND Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:0	**PS Blend State Body** Format: 3DSTATE_PS_BLEND_BODY

3DSTATE_PS

3DSTATE_PS		
Source:	RenderCS	
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 20h 3DSTATE_PS Format: OpCode
	15	**Reserved**
	14:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0Ah Excludes DWord (0,1) Format: =n Total Length - 2
1..11	351:0	**PS State Body** Format: 3DSTATE_PS_BODY

3DSTATE_PS_EXTRA

3DSTATE_PS_EXTRA		
Source: RenderCS Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 4fh 3DSTATE_PS_EXTRA Format: OpCode
	15	**Reserved**
	14:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:0	**PS Extra State Body** Format: 3DSTATE_PS_EXTRA_BODY

3DSTATE_PTBR_FREE_LIST_BASE_ADDRESS

3DSTATE_PTBR_FREE_LIST_BASE_ADDRESS

Source: PositionCS

Length Bias: 2

The 3DSTATE_PTBR_FREE_LIST_BASE_ADDRESS command programmed specifies a 4KB aligned base address to 128KB (2^{16} * 2B) of contiguous memory surface called Free-List memory. This command must be programmed only for POSH pipe and must not be programmed for render pipe. This command is context specific and must be programmed for each of the POSH enabled context. This command gets context save/restored as part of the POSH context.

POSH pipe uses Free-List memory to track the free pages (page offsets from PTBR Page Pool Base Address are stored) available for recording visibility data. Pages get consumed form the Free-List as and when the visibility data is recorded by POSH pipe and pages get added to the Free-List when freed up by the render engine on rendering.

Programming Notes

The Free List Base Address must be set to a valid page if any batch buffer is enabled for the POSH pipe.

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: 24h 3DSTATE_PTBR_FREE_LIST_BASE_ADDRESS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 01h Excludes DWord (1,2) Format: =n Total Length - 2
1..2	63:12	**Free List Base Address** Format: GraphicsAddress63-12 The Free List Base Address specifies a 4KB aligned base address for a Free-List memory.

3DSTATE_PTBR_FREE_LIST_BASE_ADDRESS		
	11:7	**Reserved** Format: MBZ
	6:0	**Free List Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the MOCS state for Free-List memory accesses using PTBR_FREE_LIST_BASE_ADDRESS.

3DSTATE_PTBR_MARKER

3DSTATE_PTBR_MARKER

Source: RenderCS

Length Bias: 2

Programming Notes

Start of Tile and End of Tile must not be set in the same command. Every Start of Tile marker programmed must have a corresponding End of Tile marker 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: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 6Ah 3DSTATE_PTBR_MARKER Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2
1	31:0	**PTBR Marker State Body** Format: 3DSTATE_PTBR_MARKER_BODY

3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS

3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS

Source: PositionCS

Length Bias: 2

The 3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS command specifies base address of the PTBR page pool allocated to HW for recording visibility data in POSH Tile Based Rendering Mode (PTBR). The size of the PTBR Page Pool Size is programmed through the MMIO register PTBR_PAGE_POOL_SIZE_REGISTER following this command. SW must always set Page Pool Restart to enabled when programming this command.
This command must be programmed only for POSH pipe and must not be programmed for render pipe. This command is context specific and must be programmed for each of the POSH enabled context. This command gets context save/restored as part of the POSH context. HW context save/restores 3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS command with Page Pool Restart set to disabled.

Programming Notes

The Page Pool Base Address must be set to a valid page if any batch buffer is enabled for the POSH pipe.

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: 21h 3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 01h Excludes DWord (1,2) Format: =n Total Length - 2
1..2	63:12	**Page Pool Base Address** Format: GraphicsAddress63-12 Page Pool Base Address specifies the 4KB aligned base address of the contiguous graphics memory allocated to HW for recording visibility data in POSH Tile Based Rendering Mode (PTBR).

3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS

	11	**Page Pool Restart** Format: Enable • SW must always set Page Pool Restart to enabled on programming 3DSTATE_PTBR_PAGE_POOL_ADDRESS command. • SW must ensure both POSH and Render pipe are flushed and synchronized prior to programing 3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS command, this ensures there aren't any pages in use on the render side from the old PTBR page pool when this command is parsed by HW.
	10:7	**Reserved** Format: MBZ
	6:0	**Page Pool Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the MOCS state for PTBR Page Pool memory accesses using Page Pool Base Address.

3DSTATE_PTBR_RENDER_LIST_BASE_ADDRESS

3DSTATE_PTBR_RENDER_LIST_BASE_ADDRESS

Source: PositionCS

Length Bias: 2

The 3DSTATE_PTBR_RENDER_LIST_BASE_ADDRESS command specifies a 4KB aligned base address for storing Render-List's in PTBR mode. Render-List is an array consisting of starting page offset's corresponding to each of the tile's visibility data. Starting page offsets stored are offsets relative to Page Pool Base Address. The number of entries (array size) in a Render-List is equivalent to the number of tiles in a Tile Pass. Each Tile Pass has a corresponding Render-List populated by POSH pipe and each entry from the Render-List is read by Render Pipe when the corresponding tile is rendered.

Programming Notes

Both POSH pipe and Render pipe use the Render List Base Address programmed in POSH pipe, SW must ensure both POSH and Render pipe are flushed and synchronized prior to programing this command.
This command must be programmed only for POSH pipe and must not be programmed for render pipe. This command is context specific and must be programmed for each of the POSH enabled context. This command gets context save/restored as part of the POSH context.

The Render List Base Address must be set to a valid page if any batch buffer is enabled for the POSH pipe.

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: 23h 3DSTATE_PTBR_RENDER_LIST_BASE_ADDRESS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 01h Excludes DWord (1,2) Format: =n Total Length - 2
1..2	63:12	**Render List Base Address**

3DSTATE_PTBR_RENDER_LIST_BASE_ADDRESS		
		Format: **GraphicsAddress63-12** Render List Base Address specifies a 4KB aligned base address for storing Render-List's in PTBR mode.
	11:7	**Reserved** Format: MBZ
	6:0	**Render List Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control start for using the Render-List Address.

3DSTATE_PTBR_TILE_PASS_INFO

3DSTATE_PTBR_TILE_PASS_INFO

Source: RenderCS, PositionCS

Length Bias: 2

The 3DSTATE_PTBR_TILE_PASS_INFO command is used to define the required parameters for a Tile Pass in PTBR mode.

Programming Notes

This command must be programmed for both POSH pipe and render pipe. This command is context specific and must be programmed for each of the POSH enabled context. This command gets context save/restored as part of the POSH context by POSH pipe and gets context save/restored as part of the render context by render pipe.

RENDER PIPE ONLY: PIPE_CONTROL DepthStallEnable and DepthCacheFlushEnable Required for the following conditions
1 . switching from a Non-PTBR mode to PTBR mode OR
2. Before the TILE_PASS_INFO.EndofTilePass marker OR
3. If Occlusion Query is enabled, this PIPE_CONTROL must also have RenderTargetCacheFlushEnable bit set.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 22h 3DSTATE_PTBR_TILE_PASS_INFO Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2 Value / Name 02h: End Of Tile Pass Marker Not Set **[Default]** 0h: End Of Tile Pass Marker Set

3DSTATE_PTBR_TILE_PASS_INFO

Programming Notes
On setting the End Of;Tile Pass Marker, SW must not program the Dword2 and Dword3 of the command and must accordingly program the Dword count in the header.

1 | 31 **End of Tile Pass**

Format:	Enable

Marker for indicating End of Tile Pass in a command sequence. This bit is always save/restored as '0' in HW CONTEXT.

Value	Name	Description
0	**[Default]**	Not an indicator of End of Tile Pass in a command sequence.
1		Indicates Start of Tile Pass in a command sequence.

Programming Notes
• A Tile Pass must begin with a "Start of Tile Pass" marker and end with "End of Tile Pass" marker. • Start of Tile Pass marker and End of Tile Pass marker must not be set in the same command • On setting the End of Tile Pass Marker, SW must not program the Dword2 and Dword3 of the command and must accordingly program the Dword count in the header.

30 **Start of Tile Pass**

Format:	Enable

Marker for indicating Start of Tile Pass in a command sequence.

Value	Name	Description
0	**[Default]**	Not an indicator of Start of Tile Pass in a command sequence.
1		Indicates Start of Tile Pass in a command sequence.

Programming Notes
• A Tile Pass must begin with a "Start of Tile Pass" marker and end with "End of Tile Pass" marker. • Start of Tile Pass marker and End of Tile Pass marker must not be set in the same command

29 **Tile Pass Flag Status**

Format:	Enable

This bit is used by hardware to context save/restore the Tile Pass Flag status.

- When set indicates context got switched out is in middle of a Tile Pass,
- when reset indicates context got switched out outside the Tile Pass.

Value	Name	Description

0	**[Default]**	Context got switched out outside the Tile Pass.
1		Context got switched out in middle of a Tile Pass.

Programming Notes
SW must never set this bit.

28:10 **Reserved**

Format:	MBZ

9:0 **Tile Count**

Format:	U10-1

Description
Tile Count indicates the number of tiles the render region is divided in to, a maximum of 128 tiles is supported by POSH pipe and a maximum of 1024 tiles by Render pipe. A tile count of zero is an special indication stating the whole render region is considered as one single tile of the dimensions of the drawing rectangle. A tile count of 1 indicates the whole render region is divided in to two tiles and tile count of 127 indicates the render region is divided in to 128 tiles. For a Tile Count greater then '0', Tile Rect Array Pointer provides the details for each of the tile's dimensions.

Value	Name	Description
0h	Default **[Default]**	The whole render region is considered as one single tile of the dimensions of the drawing rectangle.
7Fh	POSH Max Value	The render region is divided in to 128 tiles.
3FFh	Render Max Value	The render region is divided in to 1024 tiles.

Programming Notes
"Tile Count" value programmed must be same in the 3DSTATE_PTBR_TILE_PASS_INFO command programmed for "Start of Tile Pass" and for "End of Tile Pass".

2 31:6 **Tile Rect Array Pointer**

Format:	DynamicStateOffset[31:6]TILE_RECT

Specifies the 64-byte aligned offset pointing to the element zero of an array, each element is a TILE_RECT state. This offset is relative to the Dynamic State Base Address. Each TILE_RECT state is two dwords in size comprising {Y Max, X Max, Y Min, X Min} of the tile. Number of TILE_RECT states in the array is equivalent to the tile count. Example: For a Tile count of 63, 64 TILE_RECT states are programmed in consecutive memory locations constituting the array, i.e 64 consecutive qwords (qword-64bits) of data in the memory, with starting qword corresponding to the TILE_RECT state of tile0 and the last qword corresponding to tile-63.

3DSTATE_PTBR_TILE_PASS_INFO

	5:0	**Reserved** Format: MBZ
3	31:6	**Render List Pointer** Format: RenderListOffset[31:6] Specifies the 64byte offset pointing to Render-List of the current Tile Pass. This offset is relative to PTBR_RENDER_LIST_BASE_ADDRESS. Render-List is an array consisting of starting page offset's corresponding to each of the tile's visibility data. Starting page offsets stored are 16bit offsets relative to PTBR_PAGE_POOL_BASE_ADDRESS. The number of entries (array size) in a Render-List is equivalent to the tile count of the current Tile Pass.
	5:2	**Reserved** Format: MBZ
	1	**Reserved** Format: MBZ

	0	**Initialize Render List**
		Format: Enable
		On parsing this command with "Initialize Render List" set, POSH pipe allocates a page for each of the tile from the Free-List to which respective visibility data will be outputted. The page offsets corresponding to these pages are written to the Render-List. POSH pipe doesn't parse the next command in sequence until the Render-List is populated. Each entry from the Render-List is read by Render Pipe when the corresponding tile is rendered. Example: For a Tile count of 63, POSH pipe will write to 64 consecutive Words (2bytes) to the Render-List, where in word-0 belongs to Tile-0, word1 belongs to Tile-1 … word-63 belongs to Tile-63.
		This bit is ignored by RenderCS on executing 3DSTATE_PTBR_TILE_PASS_INFO command.
		Programming Notes
		SW must ensure Render-List is populated by POSH pipe before it gets read by the Render pipe. This can be achieved by using semaphores in render pipe command sequence to wait for POSH pipe to complete Render-List update.
		• SW must always set this bit to '1'.
		• SW must Initialize Render List only once in a Tile Pass.
		• SW must ensure to program the below listed commands prior to programming 3DSTATE_PTBR_TILE_PASS_INFO with "Initialize Render List" bit set.
		  • 3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS
		  • 3DSTATE_PTBR_FREE_LIST
		  • 3DSTATE_PTBR_RENDER_LIST
		This bit is always save/restored as '0' in HW CONTEXT.

3DSTATE_PTBR_TILE_SELECT

3DSTATE_PTBR_TILE_SELECT

Source: RenderCS

Length Bias: 2

The 3DSTATE_PTBR_TILE_SELECT command is programmed in render engine to define the required parameters for a tile to be rendered in POSH Tile Base Rendering mode (PTBR).
This command must be programmed only for render pipe and must not be programmed for POSH pipe. This command is context specific and must be programmed for each of the POSH enabled context. This command gets context save/restored by render engine as part of the render context.

Programming Notes

SW must ensure Render-List is populated by POSH pipe before it gets read by the Render pipe.
This can be achieved by using semaphores in render pipe command sequence to wait for POSH pipe to complete Render-List update prior to programming of 3DSTATE_PTBR_TILE_SELECT command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 6Bh 3DSTATE_PTBR_TILE_SELECT Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2
1	31:0	**PTBR Tile Select State Body** Format: 3DSTATE_PTBR_TILE_SELECT_BODY

3DSTATE_PUSH_CONSTANT_ALLOC_DS

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 of the push constants 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 Domain Shader Push Constants state must be committed prior to shaders generating thread payloads after programming the 3DSTATE_PUSH_CONSTANT_ALLOC_DS.

When gather at set shader is disabled, Push Constantcommand is committed when 3DPRIMITIVE command is parsed.
When gather at set shader is enabled, commit point on the Push Constantcommand is a 3DSTATE_BINDING_TABLE_POINTER command.

The 3DSTATE_BINDING_TABLE_POINTER must be reprogrammed after 3DSTATE_PUSH_CONST_ALLOC command and the reprogramming of the push constants prior to the next 3DPRIMITIVE if gather at set shader is enabled to ensure the new programming is committed.

DWord	Bit	Description
0 **Programming Notes:** Workaround:This command must be followed by a PIPE_CONTROL with CS Stall bit set.,	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 14h 3DSTATE_PUSH_CONSTANT_ALLOC_DS Format: OpCode
	15:8	**Reserved** Format: MBZ

3DSTATE_PUSH_CONSTANT_ALLOC_DS

	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:21	**Reserved** Format: MBZ
	20:16	**Constant Buffer Offset** Format: U5 Specifies the offset of the DS constant buffer into the URB. Value: [0,31] — Name: (0KB - 31KB) Increments of 2KB
	15:6	**Reserved** Format: MBZ
	5:0	**Constant Buffer Size** Format: U6 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: [0,32] — Name: (0KB - 32KB) Increments of 2KB

3DSTATE_PUSH_CONSTANT_ALLOC_GS

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 of the push constants 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 GeometryShader Push Constantsstate must be committed prior to shaders generating thread payloads after programming the 3DSTATE_PUSH_CONSTANT_ALLOC_GS.

When gather at set shader is disabled, Push Constantcommand is committed when 3DPRIMITIVE command is parsed.
When gather at set shader is enabled, commit point on the Push Constantcommand is a 3DSTATE_BINDING_TABLE_POINTER command.

The 3DSTATE_BINDING_TABLE_POINTER must be reprogrammed after 3DSTATE_PUSH_CONST_ALLOC command and the reprogramming of the push constants prior to the next 3DPRIMITIVE if gather at set shader is enabled to ensure the new programming is committed.

Workaround

Workaround: This command must be followed by a PIPE_CONTROL with CS Stall bit set.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 15h 3DSTATE_PUSH_CONSTANT_ALLOC_GS Format: OpCode

3DSTATE_PUSH_CONSTANT_ALLOC_GS

	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2 **Value** / **Name** / **Description** 0h / 3DSTATE_PUSH_CONSTANT_ALLOC_GS **[Default]** / Excludes DWord (0,1)
1	31:21	**Reserved** Format: MBZ
	20:16	**Constant Buffer Offset** Format: U5 Specifies the offset of the GS constant buffer into the URB. **Value** / **Name** [0,31] / (0KB - 31KB) Increments of 2KB
	15:6	**Reserved** Format: MBZ
	5:0	**Constant Buffer Size** Format: U6 Specifies the size of the GS constant buffer. This value will determine the amount of data the command stream can pre-fetch before the buffer is full. Value of zero is only valid when constants are not enabled for GS. **Value** / **Name** [0,32] / (0KB - 32KB) Increments of 2KB

3DSTATE_PUSH_CONSTANT_ALLOC_HS

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 of the push constants 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 Hull Shader Push Constants state must be committed prior to shaders generating thread payloads after programming the 3DSTATE_PUSH_CONSTANT_ALLOC_HS.

When gather at set shader is disabled, Push Constantcommand is committed when 3DPRIMITIVE command is parsed.
When gather at set shader is enabled, commit point on the Push Constantcommand is a 3DSTATE_BINDING_TABLE_POINTER command.

The 3DSTATE_BINDING_TABLE_POINTER must be reprogrammed after 3DSTATE_PUSH_CONST_ALLOC command and the reprogramming of the push constants prior to the next 3DPRIMITIVE if gather at set shader is enabled to ensure the new programming is committed.

Workaround

Workaround: This command must be followed by a PIPE_CONTROL with CS Stall bit set.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 13h 3DSTATE_PUSH_CONSTANT_ALLOC_HS

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

3DSTATE_PUSH_CONSTANT_ALLOC_PS

3DSTATE_PUSH_CONSTANT_ALLOC_PS

Source: RenderCS

Length Bias: 2

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

Programming Notes

Restriction:

- The sum of the Constant Buffer Offset and the Constant Buffer Size may not exceed the maximum value of the Constant Buffer Size.
- The sum of the constant length of the push constants 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 PixelShader Push Constantsstate must be committed prior to shaders generating thread payloads after programming the 3DSTATE_PUSH_CONSTANT_ALLOC_PS.

When gather at set shader is disabled, Push Constantcommand is committed when 3DPRIMITIVE command is parsed.
When gather at set shader is enabled, commit point on the Push Constantcommand is a 3DSTATE_BINDING_TABLE_POINTER command.

The 3DSTATE_BINDING_TABLE_POINTER must be reprogrammed after 3DSTATE_PUSH_CONST_ALLOC command and the reprogramming of the push constants prior to the next 3DPRIMITIVE if gather at set shader is enabled to ensure the new programming is committed.

Workaround

Workaround: This command must be followed by a PIPE_CONTROL with CS Stall bit set.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 16h 3DSTATE_PUSH_CONSTANT_ALLOC_PS

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	31:21	**Reserved** Format: MBZ
	20:16	**Constant Buffer Offset** Format: U5 Specifies the offset of the PS constant buffer into the URB. **Value** / **Name** [0,31] (0KB - 31KB) Increments of 2KB
	15:6	**Reserved** Format: MBZ
	5:0	**Constant Buffer Size** Format: U6 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** [0,32] (0KB - 32KB) Increments of 2KB

3DSTATE_PUSH_CONSTANT_ALLOC_VS

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 of the push constants 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 VertexShader Push Constants state must be committed prior to shaders generating thread payloads after programming the 3DSTATE_PUSH_CONSTANT_ALLOC_VS.

When executed in the POCS command stream, this command programs state for the VSR stage of the POCS pipeline. The offset & size of the VSR allocation is relative to the 32KB of URB reserved for POCS pipe Push Constant Buffers, vs. the 32KB used by the RCS pipe.

When gather at set shader is disabled, programmed constants are committed when 3DPRIMITIVE command is parsed.
When gather at set shader is enabled, commit point of the constants programmed area 3DSTATE_BINDING_TABLE_POINTER command.

The 3DSTATE_BINDING_TABLE_POINTER must be reprogrammed after 3DSTATE_PUSH_CONST_ALLOC command and the reprogramming of the push constants prior to the next 3DPRIMITIVE if gather at set shader is enabled to ensure the new programming is committed.

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

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	31:21	**Reserved** Format: MBZ
	20:16	**Constant Buffer Offset** Format: U5 Specifies the offset of the VS constant buffer into the URB. **Value** [0,31] — **Name** (0KB - 31KB) Increments of 2KB **Programming Notes** When executed from the POCS pipe, the offset is relative to theVSR_PUSH_CONSTANT_BASE (MMIO offset e518)region reserved for POCS pipe Push Constants.
	15:6	**Reserved** Format: MBZ
	5:0	**Constant Buffer Size** Format: U6 Specifies the size of the VS constant buffer. This value will determine the amount of data the command stream can pre-fetch before the buffer is full. Value of zero is only valid when constants are not enabled for VS. **Value** [0,32] — **Name** (0KB - 32KB) Increments of 2KB

3DSTATE_RASTER

3DSTATE_RASTER

Source: RenderCS

Length Bias: 2

Restriction
Restriction : When executed in the POCS command stream, this command programs the raster state for CLR and SFR stages of the POCS pipeline

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 50h 3DSTATE_RASTER Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 03h Excludes DWord (0,1) Format: =n Total Length - 2
1..4	127:0	**Raster State Body** Format: 3DSTATE_RASTER_BODY

3DSTATE_SAMPLE_MASK

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	31:0	**Sample Mask State Body** Format: 3DSTATE_SAMPLE_MASK_BODY

3DSTATE_SAMPLE_PATTERN

3DSTATE_SAMPLE_PATTERN

Source: RenderCS

Length Bias: 2

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

Programming Notes

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

Restriction : When executed in the POCS command stream, this command programs the multisample pattern state for the CLR and SFR stage of the POCS pipeline

Workaround

Workaround: This command must be followed by 32 PIPE_CONTROL commands with post sync op enabled with no CS stall bit set. This is to ensure there are no STATE_ACK cycles during the processing of the 3DSTATE_SAMPLE_PATTERN at the windower stage.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 1Ch 3DSTATE_SAMPLE_PATTERN Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 7 Excludes Dword (0,1) Format: =n Total Length - 2
1	31:28	**16x Sample3 X Offset**

		Format: U0.4 Subpixel X offset of Sample 3 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	27:24	**16x Sample3 Y Offset** Format: U0.4 Subpixel Y offset of Sample 3 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	23:20	**16x Sample2 X Offset** Format: U0.4 Subpixel X offset of Sample 2 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	19:16	**16x Sample2 Y Offset** Format: U0.4 Subpixel Y offset of Sample 2 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES_4 , _8 or _16. Range: [0,0.9375]
	15:12	**16x Sample1 X Offset** Format: U0.4 Subpixel X offset of Sample 1 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES_2, _4 , _8 or _16. Range: [0,0.9375]
	11:8	**16x Sample1 Y Offset** Format: U0.4 Subpixel Y offset of Sample 1 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	7:4	**16x Sample0 X Offset** Format: U0.4 Subpixel X offset of Sample 0 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	3:0	**16x Sample0 Y Offset**

3DSTATE_SAMPLE_PATTERN

		Format: U0.4 Subpixel Y offset of Sample 0 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
2	31:28	**16x Sample7 X Offset** Format: U0.4 Subpixel X offset of Sample 7 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	27:24	**16x Sample7 Y Offset** Format: U0.4 Subpixel Y offset of Sample 7 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	23:20	**16x Sample6 X Offset** Format: U0.4 Subpixel X offset of Sample 6 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES__8 or _16. Range: [0,0.9375]
	19:16	**16x Sample6 Y Offset** Format: U0.4 Subpixel Y offset of Sample 6 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	15:12	**16x Sample5 X Offset** Format: U0.4 Subpixel X offset of Sample 5 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	11:8	**16x Sample5 Y Offset** Format: U0.4 Subpixel Y offset of Sample 5 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES__8 or _16. Range: [0,0.9375]
	7:4	**16x Sample4 X Offset**

		Format: U0.4 Subpixel X offset of Sample 4 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES__8 or _16. Range: [0,0.9375]
	3:0	**16x Sample4 Y Offset** Format: U0.4 Subpixel Y offset of Sample 4 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
3	31:28	**16x Sample11 X Offset** Format: U0.4 Subpixel X offset of Sample 11 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	27:24	**16x Sample11 Y Offset** Format: U0.4 Subpixel Y offset of Sample 11 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES _16. Range: [0,0.9375]
	23:20	**16x Sample10 X Offset** Format: U0.4 Subpixel X offset of Sample 10 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	19:16	**16x Sample10 Y Offset** Format: U0.4 Subpixel Y offset of Sample 10 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES _16 Range: [0,0.9375]
	15:12	**16x Sample9 X Offset** Format: U0.4 Subpixel X offset of Sample 9 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	11:8	**16x Sample9 Y Offset**

3DSTATE_SAMPLE_PATTERN

		Format: U0.4 Subpixel Y offset of Sample 9 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	7:4	**16x Sample8 X Offset** Format: U0.4 Subpixel X offset of Sample 8 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	3:0	**16x Sample8 Y Offset** Format: U0.4 Subpixel Y offset of Sample 8 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
4	31:28	**16x Sample15 X Offset** Format: U0.4 Subpixel X offset of Sample 15 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	27:24	**16x Sample15 Y Offset** Format: U0.4 Subpixel Y offset of Sample 15 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	23:20	**16x Sample14 X Offset** Format: U0.4 Subpixel X offset of Sample 14 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES _16. Range: [0,0.9375]
	19:16	**16x Sample14 Y Offset** Format: U0.4 Subpixel Y offset of Sample 14 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES _16 Range: [0,0.9375]
	15:12	**16x Sample13 X Offset**

		Format: U0.4 Subpixel X offset of Sample 13 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	11:8	**16x Sample13 Y Offset** Format: U0.4 Subpixel Y offset of Sample 13 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	7:4	**16x Sample12 X Offset** Format: U0.4 Subpixel X offset of Sample 12 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES_16. Range: [0,0.9375]
	3:0	**16x Sample12 Y Offset** Format: U0.4 Subpixel Y offset of Sample 12 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
5	31:28	**8x Sample7 X Offset** Format: U0.4 Subpixel X offset of Sample 7 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	27:24	**8x Sample7 Y Offset** Format: U0.4 Subpixel Y offset of Sample 7 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	23:20	**8x Sample6 X Offset** Format: U0.4 Subpixel X offset of Sample 6 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	19:16	**8x Sample6 Y Offset** Format: U0.4

3DSTATE_SAMPLE_PATTERN

		Subpixel Y offset of Sample 6 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	15:12	**8x Sample5 X Offset** Format: U0.4 Subpixel X offset of Sample 5 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	11:8	**8x Sample5 Y Offset** Format: U0.4 Subpixel Y offset of Sample 5 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	7:4	**8x Sample4 X Offset** Format: U0.4 Subpixel X offset of Sample 4 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	3:0	**8x Sample4 Y Offset** Format: U0.4 Subpixel Y offset of Sample 4 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
6	31:28	**8x Sample3 X Offset** Format: U0.4 Subpixel X offset of Sample 3 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	27:24	**8x Sample3 Y Offset** Format: U0.4 Subpixel Y offset of Sample 3 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	23:20	**8x Sample2 X Offset** Format: U0.4 Subpixel X offset of Sample 2 relative to the UL pixel origin for 8x mode.

		Range: [0,0.9375]
	19:16	**8x Sample2 Y Offset** Format: U0.4 Subpixel Y offset of Sample 2 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	15:12	**8x Sample1 X Offset** Format: U0.4 Subpixel X offset of Sample 1 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	11:8	**8x Sample1 Y Offset** Format: U0.4 Subpixel Y offset of Sample 1 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	7:4	**8x Sample0 X Offset** Format: U0.4 Subpixel X offset of Sample 0 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	3:0	**8x Sample0 Y Offset** Format: U0.4 Subpixel Y offset of Sample 0 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
7	31:28	**4x Sample3 X Offset** Format: U0.4 Subpixel X offset of Sample 3 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]
	27:24	**4x Sample3 Y Offset** Format: U0.4 Subpixel Y offset of Sample 3 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]
	23:20	**4x Sample2 X Offset**

3DSTATE_SAMPLE_PATTERN

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

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

3DSTATE_SAMPLER_PALETTE_LOAD0

3DSTATE_SAMPLER_PALETTE_LOAD0

Source: RenderCS

Length Bias: 2

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: Opcode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: Opcode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: Opcode
	23:16	**3D Command Sub Opcode** Default Value: 02h 3DSTATE_SAMPLER_PALETTE_LOAD0 Format: Opcode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Total Length = 1 + entryCount - 2, where entryCount has a range of 1 - 256 Value: 0h — Excludes DWord (0,1) **[Default]** Value: [0,255] — Range
1..n	31:0	**Entry** Format: PALETTE_ENTRY

3DSTATE_SAMPLER_PALETTE_LOAD1

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_NONPIPELINED 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

3DSTATE_SAMPLER_PALETTE_LOAD1		
		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

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	31:0	**Sampler State Pointers State Body** Format: 3DSTATE_SAMPLER_STATE_POINTERS_BODY

3DSTATE_SAMPLER_STATE_POINTERS_GS

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	31:0	**Sampler State Pointers State Body** Format: 3DSTATE_SAMPLER_STATE_POINTERS_BODY

3DSTATE_SAMPLER_STATE_POINTERS_HS

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	31:0	**Sampler State Pointers State Body** Format: 3DSTATE_SAMPLER_STATE_POINTERS_BODY

3DSTATE_SAMPLER_STATE_POINTERS_PS

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	**Reserved**
	14:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:0	**Sampler State Pointers State Body** Format: 3DSTATE_SAMPLER_STATE_POINTERS_BODY

3DSTATE_SAMPLER_STATE_POINTERS_VS

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	31:0	**Sampler State Pointers State Body** Format: 3DSTATE_SAMPLER_STATE_POINTERS_BODY

3DSTATE_SBE

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: 04h Excludes DWord (0,1) Format: =n Total Length - 2
1..5	159:0	**SBE State Body** Format: 3DSTATE_SBE_BODY

3DSTATE_SBE_SWIZ

3DSTATE_SBE_SWIZ		
Source: RenderCS Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 51h 3DSTATE_SBE_SWIZ Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 9h Excludes DWord (0,1) Format: =n Total Length - 2
1..10	319:0	**SBE SWIZ State Body** Format: 3DSTATE_SBE_SWIZ_BODY

3DSTATE_SCISSOR_STATE_POINTERS

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	31:0	**Scissor State Pointers Body** Format: 3DSTATE_SCISSOR_STATE_POINTERS_BODY

3DSTATE_SF

3DSTATE_SF

Source: RenderCS

Length Bias: 2

Restriction

Restriction : When executed in the POCS command stream, this command programs the state for the SFR stage of the POCS pipeline.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 13h 3DSTATE_SF Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 2h Excludes DWord (0,1) Format: =n Total Length - 2
1..3	95:0	**SF State Body** Format: 3DSTATE_SF_BODY

3DSTATE_SLICE_TABLE_STATE_POINTERS

3DSTATE_SLICE_TABLE_STATE_POINTERS

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: 20h 3DSTATE_SLICE_TABLE_STATE_POINTERS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:6	**Slice Hash Table State Pointer** Format: DynamicStateOffset[31:6]SLICE_HASH_TABLE*10 Specifies the 64-byte aligned offset of the SLICE_HASH_TABLE. This offset is relative to the **Dynamic State Base Address**.
	5:1	**Reserved** Format: MBZ
	0	**Slice Hash State Pointer Valid** Format: Enable This bit, if set, indicates that the SLICE_HASH_TABLE pointer has changed and new state needs to be fetched. Value: 0h — Name: Disable Value: 1h — Name: Enable

3DSTATE_SO_BUFFER

3DSTATE_SO_BUFFER

Source: RenderCS

Length Bias: 2

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

Workaround
Workaround: This command must be followed by a PIPE_CONTROL with CS Stall bit set.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 18h 3DSTATE_SO_BUFFER Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 6h Excludes DWord (0,1) Format: =n Total Length - 2
1	31	**SO Buffer Enable** Format: Enable If set, stream output to SO Buffer is enabled, , if 3DSTATE_STREAMOUT::SO Function ENABLE is also enabled.. If clear, the SO Buffer is considered "not bound" and effectively treated as a zero-length buffer for the purposes of SO output and overflow detection. If an enabled stream's Stream to Buffer Selects includes this buffer it is by definition an overflow condition. That stream will cause no writes to occur, and only SO_PRIM_STORAGE_NEEDED[<stream>] will increment.

3DSTATE_SO_BUFFER

	30:29	**SO Buffer Index** Format: U2 Specifies which of the four SO Buffers is being defined.
	28:22	**SO Buffer Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for the SO buffer.
	21	**Stream Offset Write Enable** Format: Enable When set, this field allows the hardware to write SO_WRITE_OFFSET[Buffer#] as specified in the Stream Offset field. **Programming Notes** The field is operates irrespective of whether SO Buffer Enable is set or clear.
	20	**Stream Output Buffer Offset Address Enable** Format: Enable When set, this field allows the hardware to read/write the stream output buffer offset as specified in the "Stream Output Buffer Offset Address" field. **Programming Notes** The field is operates irrespective of whether SO Buffer Enable is set or clear.
	19:12	**Reserved** Format: MBZ
	11:0	**Reserved** Format: MBZ
2..3	63:2	**Surface Base Address** Format: GraphicsAddress63-2 This field specifies the starting address of the buffer in Graphics Memory.
	1:0	**Reserved** Format: MBZ
4	31:30	**Reserved** Format: MBZ
	29:0	**Surface Size** Format: U30-1 This field specifies the size of buffer in number DWords minus 1 of the buffer in Graphics Memory.

3DSTATE_SO_BUFFER		
5..6	63:2	**Stream Output Buffer Offset Address** Format: GraphicsAddress63-2 This field specifies the starting address of the buffer in Graphics Memory where the Stream Output Buffer Offset is stored when all the data has been written. It is also used to fetch the stream Output buffer Offset when needed.
	1:0	**Reserved** Format: MBZ
7	31:0	**Stream Offset** This field specifies the Offset in stream output buffer to start at, or whether to append to the end of an existing buffer. The Offset must be DWORD aligned. If Stream Offset is equal to 0xFFFFFFFF then load the value at the Stream Output Buffer Offset address into SO_WRITE_OFFSET[Buffer#]. Otherwise, SO_WRITE_OFFSET[Buffer#] = Stream Offset.

3DSTATE_SO_DECL_LIST

3DSTATE_SO_DECL_LIST

Source: RenderCS

Length Bias: 2

Workaround
Workaround: This command must be followed by a PIPE_CONTROL with CS Stall bit set.,

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 17h 3DSTATE_SO_DECL_LIST Format: OpCode
	15:9	**Reserved** Format: MBZ
	8:0	**DWord Length** Format: =n Total Length - 2 Value / Name / Description [1,257] / Excludes DWORD (0,1) 0-128 Entries / Value = 2 * (# of SO_DECL quads) + 1
1	31:16	**Reserved** Format: MBZ
	15:12	**Stream to Buffer Selects [3]** Format: U4 bitmask Identifies to which SO Buffers stream 3 outputs. See Stream To Buffer Selects [0] field description. Value / Name 1xxxb / SO Buffer 3 x1xxb / SO Buffer 2

		xx1xb: SO Buffer 1 xxx1b: SO Buffer 0
	11:8	**Stream to Buffer Selects [2]** Format: U4 bitmask Identifies to which SO Buffers stream 2 outputs. See Stream To Buffer Selects [0] field description. 1xxxb: SO Buffer 3 x1xxb: SO Buffer 2 xx1xb: SO Buffer 1 xxx1b: SO Buffer 0
	7:4	**Stream to Buffer Selects [1]** Format: U4 bitmask Identifies to which SO Buffers stream 1 outputs. See Stream To Buffer Selects [0] field description. 1xxxb: SO Buffer 3 x1xxb: SO Buffer 2 xx1xb: SO Buffer 1 xxx1b: SO Buffer 0
	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). 1xxxb: SO Buffer 3 x1xxb: SO Buffer 2 xx1xb: SO Buffer 1 xxx1b: SO Buffer 0
2	31:24	**Num Entries [3]** Format: U8 #entries Specifies the number of valid SO_DECL entries for Stream 3. (See notes in Num Entries [0] field description).

3DSTATE_SO_DECL_LIST

		[0,128] entries
	23:16	**Num Entries [2]** Format: U8 #entries Specifies the number of valid SO_DECL entries for Stream 2. (See notes in Num Entries [0] field description). **Value** / **Name** [0,128] entries
	15:8	**Num Entries [1]** Format: U8 #entries Specifies the number of valid SO_DECL entries for Stream 1. (See notes in Num Entries [0] field description). **Value** / **Name** [0,128] entries
	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** [0,128] entries
3..n	63:0	**Entry** Format: SO_DECL_ENTRY

3DSTATE_STENCIL_BUFFER

3DSTATE_STENCIL_BUFFER

Source: RenderCS

Length Bias: 2

Description

This command sets the surface state of the separate stencil buffer, delivered as a pipelined state command. However, the state change pipelining isn't completely transparent (see restriction below).

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

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).

The stencil buffer is always Tile-W

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 | Name

3DSTATE_STENCIL_BUFFER

		3h: Excludes Dword (0,1) **[Default]**
1	31	**Stencil Buffer Enable** Format: U1 When set indicates that there is a valid stencil buffer. **Programming Notes** This bit should be "0" if Depth buffer surface format is D16_UNORM OR Depth buffer surface type is NULL.
	30:29	**Reserved** Format: MBZ
	28:22	**Stencil Buffer Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for the stencil buffer.
	21	**Corner Texel Mode** Format: Enable This field, when ENABLED, indicates when a surface is using corner texel-mode for stencil surface. This bit changes how the size of each MIP when calculating the offset within a surface.
	20:17	**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).
2..3	63:0	**Surface Base Address** Format: GraphicsAddress[63:0]Stencil_Buffer This field specifies the address of the buffer in Graphics Memory.

Value	Name	Description
0h	Disable **[Default]**	Corner Texel Mode is disabled by default.
1h	Enable	Corner Texel Mode is enabled.

Value	Name	Description
[127, 1FFFFh]		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).

3DSTATE_STENCIL_BUFFER

Programming Notes

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

4	31:15	**Reserved** Format: MBZ
	14:0	**Surface QPitch** Format: U15[16:2]

Description

This field specifies the distance in rows between array slices. It is used only in the following cases:

- **Surface Array** is enabled *OR*
- **Number of Mulitsamples** is not NUMSAMPLES_1 and **Multisampled Surface Storage Format** set to MSFMT_MSS *OR*
- **Surface Type** is SURFTYPE_CUBE

The interpretation of this field is dependent on **Surface Type** as follows:

- SURFTYPE_1D: distance in *pixels* between array slices
- SURFTYPE_2D/CUBE: distance in *rows* between array slices
- SURFTYPE_3D: distance in *rows* between R-slices

Format:

```
QPitch[16:2]
```

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

Programming Notes

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

3DSTATE_STREAMOUT

3DSTATE_STREAMOUT

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: 3h Excludes DWord (0,1) Format: =n Total Length - 2
1..4	127:0	**Streamout State Body** Format: 3DSTATE_STREAMOUT_BODY

3DSTATE_SUBSLICE_HASH_TABLE

3DSTATE_SUBSLICE_HASH_TABLE

Source: RenderCS

Length Bias: 2

Description

Programmable DualSubSlice hashing control and tables. First DW indicates the mode how the last 4DW are configured and selects the programmable dualsubslice hashing mode for each slice.

Programming Notes

All slice references are physical slice. Instruction must be programmed based on which slices and subslices are enabled.

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: 1Fh 3DSTATE_SUBSLICE_HASH_TABLE Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2 Value: 4h — Name: Excludes DWord (0,1) **[Default]**
1	31:30	**TableMode** Format: U2 Value: 0h — Name: Single table **[Default]** — Description: DW2-5 is Table[0] - a single 2-way 128 entry [Y][X] table.

3DSTATE_SUBSLICE_HASH_TABLE

		1h: Dual tables — DW2-3 is 'Table[0]' and is 2-way 64 entry [Y][X] table. DW4-5 is 'Table[1]' and is 2-way 64 entry [Y][X] table.
	29:16	**Reserved** Format: MBZ
	15:0	**SliceHashCtrl** PerSlice[7:0]SliceHashControl
2..5	127:64	**64 Entry 2-way Table[1]** Exists If: Instruction[3DSTATE_SUBSLICE_HASH_TABLE][TableMode]==Dual Format: **pixel_hash_table_1bit_64entry**
	127:0	**128 Entry 2-way Table[0]** Exists If: Instruction[3DSTATE_SUBSLICE_HASH_TABLE][TableMode]==Single Format: **pixel_hash_table_1bit_128entry**
	63:0	**64 Entry 2-way Table[0]** Exists If: Instruction[3DSTATE_SUBSLICE_HASH_TABLE][TableMode]==Dual Format: **pixel_hash_table_1bit_64entry**

3DSTATE_TE

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** Format: =n Total Length - 2 **Value** / **Name** 2h: Excludes DWord (0,1) **[Default]**
1..3	95:0	**TE State Body** Format: 3DSTATE_TE_BODY

3DSTATE_URB_CLEAR

3DSTATE_URB_CLEAR

Source: RenderCS

Length Bias: 2

The 3DSTATE_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 not a part of context save/restore.

Workaround

Workaround: This command must be followed by a PIPE_CONTROL with CS Stall bit set.,

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 1Dh 3DSTATE_URB_CLEAR Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:30	**Reserved** Format: MBZ
	29:16	**URB Clear Length** This field specifies the number of 256b entries in the URB to be cleared to zero.
	15	**Reserved**

3DSTATE_URB_CLEAR			
		Format:	MBZ
	14:0	**URB Address** Format: URBAddress[19:5] 256b aligned This field specifies Bits 19:5 of the URB Address	

3DSTATE_URB_DS

3DSTATE_URB_DS

Source: RenderCS

Length Bias: 2

Description

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

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

Programming Notes

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 32h 3DSTATE_URB_DS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:0	**URB DS State Body** Format: 3DSTATE_URB_DS_BODY

3DSTATE_URB_GS

3DSTATE_URB_GS

Source: RenderCS

Length Bias: 2

Description

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

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

Programming Notes

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 33h 3DSTATE_URB_GS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:0	**URB GS State Body** Format: 3DSTATE_URB_GS_BODY

3DSTATE_URB_HS

3DSTATE_URB_HS

Source: RenderCS

Length Bias: 2

Description

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

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

Programming Notes

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 31h 3DSTATE_URB_HS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:0	**URB HS State Body** Format: 3DSTATE_URB_HS_BODY

3DSTATE_URB_VS

3DSTATE_URB_VS

Source: RenderCS, PositionCS

Length Bias: 2

Description

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

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

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

When executed from the POCS command stream, state for the VSR stage is set. The VSR URB region shall never overlap any other URB region. As POCS and RCS command streams are not implicitly synchronized, if POCS is used SW shall reserve a region of the URB for use by VSR. Both pipelines must be flushed and synchronized before expanding the VSR URB region such that the new VSR URB region overlaps URB space previously used by the render pipeline, or the URB space to be used by the render pipeline overlaps the previous VSR URB region. The POCS command stream will only execute 3DSTATE_URB_VS command with respect to URB programming for the POCS 3D pipeline. 3DSTATE_URB_HS, 3DSTATE_URB_DS and 3DSTATE_URB_GS commands are ignored by the POCS command stream.

Programming Notes

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 30h 3DSTATE_URB_VS Format: OpCode

3DSTATE_URB_VS		
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:0	**URB VS State Body** Format: 3DSTATE_URB_VS_BODY

3DSTATE_VERTEX_BUFFERS

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).
When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 03h 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: 08h 3DSTATE_VERTEX_BUFFERS

3DSTATE_VERTEX_BUFFERS

		Format: Opcode
	15	**Reserved**
	14:8	**Reserved**
	7:0	**DWord Length** Format: =n n = 4b-1 (where b = # of buffer states included) **Value** / **Name** 3 / DWORD_COUNT_n **[Default]** [3,131] / 1-33 Buffers
1..n	127:0	**Vertex Buffer State** Format: VERTEX_BUFFER_STATE

3DSTATE_VERTEX_ELEMENTS

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. (E.g.. 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.
When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline.

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_PIPELINED Format: Opcode

3DSTATE_VERTEX_ELEMENTS

	23:16	**3D Command Sub Opcode** Default Value: 09h 3DSTATE_VERTEX_ELEMENTS Format: Opcode
	15	**Reserved**
	14:8	**Reserved**
	7:0	**DWord Length** Format: =n Vertex Element Count = (DWord Count + 1) / 2 **Value** / **Name** / **Description** 1 / DWORD_COUNT_n **[Default]** / excludes DWords 0,1 [1,67] / Range / 1-34 Elements
1..n	63:0	**Element** Format: **VERTEX_ELEMENT_STATE** []

3DSTATE_VF_COMPONENT_PACKING

3DSTATE_VF_COMPONENT_PACKING

Source: RenderCS

Length Bias: 2

This command is used to specify, separately for Vertex Elements [0-31], which post-conversion, 32-bit components are "enabled" to be stored in the URB , and which are "disabled" (not stored). 128 per-component enable bits are provided. Disabling all four components for a given Vertex Element will result in no data stored for that element. Note that any insertion of SGVs (3DSTATE_VF_SGVS) is performed before the packing operation. The **Component Packing Enable** bit (3DSTATE_VF) controls the overall packing process. If that bit is set, the packing process is enabled and the bit mask provided in this command is used to control which components are stored. If that bit is clear, the packing process is disabled - all four components of "valid" Vertex Elements will be stored.

Programming Notes

Programming Restrictions:

- The Vertex Elements referenced in this command correspond to the first 32 VERTEX_ELEMENT structures passed in 3DSTATE_VERTEX_ELEMENTS. A Vertex Element must be marked as "Valid" via 3DSTATE_VERTEX_ELEMENTS or be an SGV or an element between the last valid element and the last SGV in order for the corresponding Component Enable bits of this command to be utilized.
- No enable bits are provided for Vertex Elements [32-33], and therefore no packing is performed on these elements (if Valid, all 4 components are stored).
- If a Vertex Element has Edge Flag Enable set no packing is performed for that element and the corresponding packing state is ignored.
- Component packing is probably only useful for SIMD8 VS thread execution.

At least one component of one "valid" Vertex Element must be enabled.

When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline.

Software shall enable all components (XYZW) for any and all VERTEX_ELEMENTs associated with a 256-bit SURFACE_FORMAT. It is INVALID to disable any components in these cases.

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

3DSTATE_VF_COMPONENT_PACKING

		Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 55h 3DSTATE_VF_COMPONENT_PACKING Format: OpCode
	15	**Reserved**
	14:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 3h Excludes DWord (0,1) Format: =n Total Length - 2
1..4	127:0	**VF Component Packing State Body** Format: 3DSTATE_VF_COMPONENT_PACKING_BODY

3DSTATE_VF

3DSTATE_VF			
Source:	RenderCS		
Length Bias:	2		
Description			
This command is used to set various state variables in the VF stage.			
The use of the component packing mask is specified via 3DSTATE_VF_COMPONENT_PACKING			
Programming Notes			
When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline.			
DWord	**Bit**	**Description**	
0	31:29	**Command Type**	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	**Command SubType**	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	**3D Command Opcode**	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	**3D Command Sub Opcode**	
		Default Value:	0Ch 3DSTATE_VF
		Format:	OpCode
	15	**Reserved**	
		Format:	MBZ
	14	**Reserved**	
		Format:	MBZ
	13	**Reserved**	
		Format:	MBZ
	12	**Reserved**	
		Format:	MBZ
	11	**VertexID Offset Enable**	
		Format:	Enable
		If ENABLED, the VertexID value optionally inserted into the vertex data is offset by StartVertexLocation (SEQUENTIAL draws) or BaseVertexLocation (RANDOM draws). If DISABLED,	

		VertexID is not offset by these values and instead is always 0-based
	10	**Sequential Draw Cut Index Enable** Format: Enable If ENABLED, vertex indices in SEQUENTIAL 3DPRIMITIVE commands are compared to the Cut Index (specified below). When the vertex index matches the Cut Index any previous topology is terminated. If DISABLED, vertex indices are not compared to the Cut Index. This field can only be enabled for certain primitive topology types. Refer to the table later in this section for details.
	9	**Component Packing Enable** Format: Enable If ENABLED, vertex element component packing (as specified by 3DSTATE_VF_COMPONENT_PACKING) is performed before vertices are written into the URB. If DISABLED, no component packing is performed - all components of valid vertex elements will be stored in the URB.
	8	**Indexed Draw Cut Index Enable** Format: Enable If ENABLED, vertex indices in RANDOM 3DPRIMITIVE commands are compared to the Cut Index (specified below). When the vertex index matches the Cut Index any previous topology is terminated. If DISABLED, vertex indices are not compared to the Cut Index and are used strictly as indices into vertex buffers. This field can only be enabled for certain primitive topology types. Refer to the table later in this section for details.
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:0	**VF State Body** Format: 3DSTATE_VF_BODY

3DSTATE_VF_INSTANCING

3DSTATE_VF_INSTANCING

Source: RenderCS

Length Bias: 2

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

Programming Notes

When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 49h 3DSTATE_VF_INSTANCING Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2 Value / Name 1h: Excludes DWord (0,1) **[Default]** 43h: Context Restore
1..2	63:0	**VF Instancing State Body** Format: 3DSTATE_VF_INSTANCING_BODY

3DSTATE_VF_SGVS_2

3DSTATE_VF_SGVS_2

Source: RenderCS

Length Bias: 2

This command is used to control the insertion of the Extended Parameter (XP0-2) System-Generated Values (SGVs) into an input Vertex URB Entry (VUE) (available as input to a VS thread). The insertions are individually-controlled. The insertion locations are specified as 128-bit element locations (starting at the beginning of the VUE) and 32-bit component within those specified elements. The SGV values can be inserted either (a) within a valid vertex element (in which case the value overwrites the value specified via 3DSTATE_VERTEX_ELEMENTS) or (b) beyond the last valid vertex element written to the URB. This permits some orthogonality between the programming of vertex elements (which typically is known at draw time) and programming of SGV insertion (which is associated with the shader). There are some restrictions however (see below). If an SGV is inserted beyond the last valid vertex element, zeroes are first inserted in the VUE after the last valid vertex element up to and including the vertex element receiving an SGV. If both of the SGVs are enabled for insertion, the zeroes will extend to the last vertex element receiving and SGV. Then the SGV(s) are inserted.
The sources for these SGV values are derived from 3DPRIMITIVE command parameters. Controls in the 3DPRIMITIVE command determine whether (a) the parameters are directly defined via in-line command DWords, (b) the parameters are indirectly specified by command stream registers, or (c) the parameters are not included in the 3DPRIMITIVE command and therefore default to 0. Refer to the 3DPRIMITIVE command description for details on these different cases.
The states included in this command are used to (a) enable/disable specific XP* SGV insertions and (b) for XP0 and XP1, specify which 3DPRIMITIVE parameters are used to source the inserted SGV value.

The insertion of SGV values occurs before any component packing (3DSTATE_VF_COMPONENT_PACKING). Therefore the Element Offsets and Component Numbers specified in this command refer to the pre-packed data, following 3DSTATE_VERTEX_ELEMENT processing.

Programming Notes

Programming Restrictions:

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

When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline.

DWord	Bit	Description
0	31:29	**Command Type**

3DSTATE_VF_SGVS_2

		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: 56h 3DSTATE_VF_SGVS_2 Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 1h Excludes DWord (0,1) Format: =n Total Length - 2
1..2	63:0	**VF SGVS 2 State Body** Format: 3DSTATE_VF_SGVS_2_BODY

3DSTATE_VF_SGVS

3DSTATE_VF_SGVS

Source: RenderCS

Length Bias: 2

Description

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

The insertion of SGV values occurs before any component packing (3DSTATE_VF_COMPONENT_PACKING). Therefore the Element Offsets and Component Numbers specified in this command refer to the pre-packed data, following 3DSTATE_VERTEX_ELEMENT processing.

Programming Notes

Programming Restrictions:

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

When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode

3DSTATE_VF_SGVS

	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: 4Ah 3DSTATE_VF_SGVS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:0	**VF SGVS State Body** Format: 3DSTATE_VF_SGVS_BODY

3DSTATE_VF_STATISTICS

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.

Programming Notes

When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: Opcode
	28:27	**Command SubType** Format: Opcode Value 1h: GFXPIPE_SINGLE_DW **[Default]**
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: Opcode GFXPIPE[28:27 = 1h, 26:24 = 0h, 23:16 = 0Bh] (Pipelined, Single DWord)
	23:16	**3D Command Sub Opcode** Default Value: 0Bh 3DSTATE_VF_STATISTICS Format: Opcode GFXPIPE[28:27 = 1h, 26:24 = 0h, 23:16 = 0Bh] (Pipelined, Single DWord)
	15:1	**Reserved** Format: MBZ
	0	**Statistics Enable** Format: Enable 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. **Programming Notes** When a 3DPRIMITIVE command with POSH Enable set is executed from the RCS command

3DSTATE_VF_STATISTICS		
		stream, VF statistics gathering is inhibited for that command.

3DSTATE_VF_TOPOLOGY

3DSTATE_VF_TOPOLOGY

Source: RenderCS

Length Bias: 2

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

Programming Notes

When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline, and only the following topologies are valid: 3DPRIM_TRILIST, 3DPRIM_TRISTRIP, 3DPRIM_TRISTRIP_REV.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 4Bh 3DSTATE_VF_TOPOLOGY Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:0	**VF Topology State Body** Format: 3DSTATE_VF_TOPOLOGY_BODY

3DSTATE_VIEWPORT_STATE_POINTERS_CC

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_CC Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:0	**Viewport State Pointers CC State Body** Format: 3DSTATE_VIEWPORT_STATE_POINTERS_CC_BODY

3DSTATE_VIEWPORT_STATE_POINTERS_SF_CLIP

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.

Restriction

Restriction : When executed in POCS command stream, this programs viewport state of the POCS pipeline.

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	31:0	**Viewport State Pointers SF Clip State Body** Format: 3DSTATE_VIEWPORT_STATE_POINTERS_SF_CLIP_BODY

3DSTATE_VS

3DSTATE_VS

Source: RenderCS, PositionCS

Length Bias: 2

Description

This command specifies most of the state used by the Vertex Shader (VS) stage.

Programming Notes

When executed in the POCS command stream, this command programs state for the VSR stage of the POCS pipeline. When executed in the RCS command stream, this command programs state for the VS stage of the RCS pipeline.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 10h 3DSTATE_VS Format: OpCode
	15	**Reserved**
	14:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 7h Excludes DWord (0,1) Format: =n Total Length - 2
1..8	255:0	**VS State Body** Format: 3DSTATE_VS_BODY

3DSTATE_WM_CHROMAKEY

3DSTATE_WM_CHROMAKEY

Source: RenderCS

Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 4Ch 3DSTATE_WM__CHROMAKEY Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**Dword Length** Default Value: 0h Excludes Dword (0,1) Format: =n Total Length - 2
1	31:0	**WM Chromakey State Body** Format: 3DSTATE_WM_CHROMAKEY_BODY

3DSTATE_WM_DEPTH_STENCIL

3DSTATE_WM_DEPTH_STENCIL		
Source: RenderCS Length Bias: 2		
This command replaces the indirect state DEPTH_STENCIL_STATE with an inline state command.		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 4Eh 3DSTATE_WM_DEPTH_STENCIL Format: OpCode
	15:13	**Reserved** Format: MBZ
	12:8	**Reserved** Format: MBZ
	7:0	**Dword Length** Format: =n Total Length - 2 Value: 02h — Name: Excludes Dword (0,1) **[Default]**
1..3	95:0	**WM Depth Stencil State Body** Format: 3DSTATE_WM_DEPTH_STENCIL_BODY

3DSTATE_WM

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: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:0	**WM State Body** Format: 3DSTATE_WM_BODY

3DSTATE_WM_HZ_OP

3DSTATE_WM_HZ_OP

Source: RenderCS

Length Bias: 2

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

Programming Notes

As this command generates an implicit rectangle, SW must make sure any MMIO register writes following WM_HZ_OP must be preceded by **PIPE_CONTROL** with **Command Streamer Stall Enable** bit set.

3DSTATE_DRAWING_RECTANGLE must be programmed such that it does not clip the HZ_OP command's rectangle. See programming notes for X/Y Min and X/Y Max below.
3DSTATE_DRAWING_RECTANGLE command must come before 3DSTATE_WM_HZ_OP in the command buffer
Caution: There is a difference in how X/Y coordinates are interpreted by 3DSTATE_DRAWING_RECTANGLE vs. 3DSTATE_WM_HZ_OP.
HZ_OP rectangle parameters are exclusive on max side, for example to have 8x4 rectangle we would program X Min = 0, Y Min = 0, X Max = 8, Y Max = 4.
Draw Rectangle parameters are inclusive on max side, meaning, for 8x4 rectangle the values would be X Min = 0, Y Min = 0, X Max = 7, Y Max = 3.

3DSTATE_MULTISAMPLE packet must be used prior to this packet to change the Number of Multisamples. This packet must not be used to change Number of Multisamples in a rendering sequence.
3DSTATE_RASTER if used must be programmed prior to using this packet.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 52h 3DSTATE_WM_HZ_OP Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**Dword Length** Format: =n

3DSTATE_WM_HZ_OP		
		Total Length - 2 **Value** / **Name** 03h / Excludes Dword (0,1) **[Default]**
1..4	127:0	**WM HZ OP State Body** Format: 3DSTATE_WM_HZ_OP_BODY

A64 Byte Scattered Read MSD

MSD1R_A64_BS - A64 Byte Scattered Read MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Scattered R/W
Group: Byte Scattered R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 10h Format: Opcode A64 Scattered Read message
	13	**Invalidate After Read** Format: MDC_IAR Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12	**SIMD Mode** Format: MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)
	11:10	**Data Elements** Format: MDC_A64_DS Specifies the number of data elements to be read or written
	9:8	**A64 Scattered Message Subtype** Default Value: 0h Format: Opcode

MSD1R_A64_BS - A64 Byte Scattered Read MSD		
		Byte Read/Write subtype
	7:0	**Binding Table Index** Format: **MDC_STATELESS** Specifies the message is stateless

A64 Byte Scattered Write MSD

MSD1W_A64_BS - A64 Byte Scattered Write MSD

Source:	EuSubFunctionDataPort1
Length Bias:	1
Family:	Scattered R/W
Group:	Byte Scattered R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 1Ah Format: Opcode A64 Scattered Write message
	13	**Reserved** Format: MBZ Ignored
	12	**SIMD Mode** Format: MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)

MSD1W_A64_BS - A64 Byte Scattered Write MSD

	11:10	**Data Elements** Format: **MDC_A64_DS** Specifies the number of data elements to be read or written
	9:8	**A64 Scattered Message Subtype** Default Value: 0h Format: Opcode Byte Read/Write subtype
	7:0	**Binding Table Index** Format: **MDC_STATELESS** Specifies the message is stateless

A64 Dword Scattered Read MSD

MSD1R_A64_DWS - A64 Dword Scattered Read MSD

Source:	EuSubFunctionDataPort1
Length Bias:	1
Family:	Scattered R/W
Group:	DW Scattered R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 10h Format: Opcode A64 Scattered Read message
	13	**Invalidate After Read** Format: MDC_IAR Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12	**SIMD Mode** Format: MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)

MSD1R_A64_DWS - A64 Dword Scattered Read MSD

	11:10	**Data Elements** Format: **MDC_A64_DS** Specifies the number of data elements to be read or written
	9:8	**A64 Scattered Message Subtype** Default Value: 1h Format: Opcode Dword Read/Write subtype
	7:0	**Binding Table Index** Format: **MDC_STATELESS** Specifies the message is stateless

A64 Dword Scattered Write MSD

MSD1W_A64_DWS - A64 Dword Scattered Write MSD

Source:	EuSubFunctionDataPort1
Length Bias:	1
Family:	Scattered R/W
Group:	DW Scattered R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 1Ah Format: Opcode A64 Scattered Write message
	13	**Reserved** Format: MBZ Ignored
	12	**SIMD Mode** Format: MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)

MSD1W_A64_DWS - A64 Dword Scattered Write MSD

	11:10	**Data Elements** Format: **MDC_A64_DS** Specifies the number of data elements to be read or written
	9:8	**A64 Scattered Message Subtype** Default Value: 1h Format: Opcode Dword Read/Write subtype
	7:0	**Binding Table Index** Format: **MDC_STATELESS** Specifies the message is stateless

A64 Dword Untyped Atomic Float Binary with Return Data Operation MSD

MSD1R_A64_DWAF2 - A64 Dword Untyped Atomic Float Binary with Return Data Operation MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Dword Untyped Atomic Float Binary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 1Dh Format: Opcode A64 Untyped Atomic Float Operation message
	13	**Return Data Control** Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.

MSD1R_A64_DWAF2 - A64 Dword Untyped Atomic Float Binary with Return Data Operation MSD

	12	**SIMD Mode** Format: **MDC_SM2S** Only SIMD8 operations are supported.
	11	**Data Width** Default Value: 0h Format: Opcode Operations are on 32-bit floats.
	10:8	**Atomic Float Operation** Format: **MDC_FOP2** Specifies the atomic float operation to be performed.
	7:0	**Binding Table Index** Format: **MDC_STATELESS** Specifies the message is stateless

A64 Dword Untyped Atomic Float Binary Write Only Operation MSD

MSD1W_A64_DWAF2 - A64 Dword Untyped Atomic Float Binary Write Only Operation MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Dword Untyped Atomic Float Binary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 1Dh Format: Opcode A64 Untyped Atomic Float Operation message
	13	**Return Data Control** Default Value: 0h Format: Opcode Specifies that no return data is sent back to the thread.

MSD1W_A64_DWAF2 - A64 Dword Untyped Atomic Float Binary Write Only Operation MSD

	12	**SIMD Mode** Format: **MDC_SM2S** Only SIMD8 operations are supported.
	11	**Data Width** Default Value: 0h Format: Opcode Operations are on 32-bit floats.
	10:8	**Atomic Float Operation Type** Format: **MDC_FOP2** Specifies the atomic float operation to be performed.
	7:0	**Binding Table Index** Format: **MDC_STATELESS** Specifies the message is stateless

A64 Dword Untyped Atomic Float Ternary with Return Data Operation MSD

MSD1R_A64_DWAF3 - A64 Dword Untyped Atomic Float Ternary with Return Data Operation MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Dword Untyped Atomic Float Ternary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 1Dh Format: Opcode A64 Untyped Atomic Float Operation message

MSD1R_A64_DWAF3 - A64 Dword Untyped Atomic Float Ternary with Return Data Operation MSD

	13	**Return Data Control** Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2S Only SIMD8 operations are supported.
	11	**Data Width** Default Value: 0h Format: Opcode Operations are on 32-bit floats.
	10:8	**Atomic Float Operation** Format: MDC_FOP3 Specifies the atomic float operation to be performed.
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Dword Untyped Atomic Float Ternary Write Only Operation MSD

MSD1W_A64_DWAF3 - A64 Dword Untyped Atomic Float Ternary Write Only Operation MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Dword Untyped Atomic Float Ternary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 1Dh Format: Opcode A64 Untyped Atomic Float Operation message

MSD1W_A64_DWAF3 - A64 Dword Untyped Atomic Float Ternary Write Only Operation MSD

	13	**Return Data Control** Default Value: 0h Format: Opcode Specifies that no return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2S Only SIMD8 operations are supported.
	11	**Data Width** Default Value: 0h Format: Opcode Operations are on 32-bit floats.
	10:8	**Atomic Float Operation** Format: MDC_FOP3 Specifies the atomic float operation to be performed.
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Dword Untyped Atomic Integer Binary with Return Data Operation MSD

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

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Dword Untyped Atomic Integer Binary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF The message forbids a header
	18:14	**Message Type** Default Value: 12h Format: Opcode A64 Untyped Atomic Integer Operation message

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

	13	**Return Data Control** Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	**Data Width** Default Value: 0h Format: Opcode Operations are on 32-bit integers
	11:8	**Atomic Integer Operation** Format: MDC_AOP2 Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Dword Untyped Atomic Integer Binary Write Only Operation MSD

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

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Dword Untyped Atomic Integer Binary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF The message forbids a header
	18:14	**Message Type** Default Value: 12h Format: Opcode A64 Untyped Atomic Integer Operation message

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

	13	**Return Data Control** Default Value: 0h Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Data Width** Default Value: 0h Format: Opcode Operations are on 32-bit integers
	11:8	**Atomic Integer Operation** Format: MDC_AOP2 Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Dword Untyped Atomic Integer Ternary with Return Data Operation MSD

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

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Dword Untyped Atomic Integer Ternary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF The message forbids a header
	18:14	**Message Type** Default Value: 12h Format: Opcode A64 Untyped Atomic Integer Operation message

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

	13	**Return Data Control** Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	**Data Width** Default Value: 0h Format: Opcode Operations are on 32-bit integers
	11:8	**Atomic Integer Operation** Format: MDC_AOP3 Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Dword Untyped Atomic Integer Ternary Write Only Operation MSD

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

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Dword Untyped Atomic Integer Ternary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF The message forbids a header
	18:14	**Message Type** Default Value: 12h Format: Opcode A64 Untyped Atomic Integer Operation message

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

	13	**Return Data Control** Default Value: 0h Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Data Width** Default Value: 0h Format: Opcode Operations are on 32-bit integers
	11:8	**Atomic Integer Operation** Format: MDC_AOP3 Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Dword Untyped Atomic Integer Unary with Return Data Operation MSD

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

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Dword Untyped Atomic Integer Unary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF The message forbids a header
	18:14	**Message Type** Default Value: 12h Format: Opcode A64 Untyped Atomic Integer Operation message

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

	13	**Return Data Control** Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	**Data Width** Default Value: 0h Format: Opcode Operations are on 32-bit integers
	11:8	**Atomic Integer Operation** Format: MDC_AOP1 Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Dword Untyped Atomic Integer Unary Write Only Operation MSD

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

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Dword Untyped Atomic Integer Unary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF The message forbids a header
	18:14	**Message Type** Default Value: 12h Format: Opcode A64 Untyped Atomic Integer Operation message

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

	13	**Return Data Control** Default Value: 0h Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Data Width** Default Value: 0h Format: Opcode Operations are on 32-bit integers
	11:8	**Atomic Integer Operation** Format: MDC_AOP1 Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Hword Block Read MSD

MSD1R_A64_HWB - A64 Hword Block Read MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Block R/W
Group: HW Block R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 14h Format: Opcode A64 Oword Block Read message
	13	**Invalidate After Read**

MSD1R_A64_HWB - A64 Hword Block Read MSD

		Format: MDC_IAR Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12:11	**A64 Block Message Subtype** Default Value: 3h Format: Opcode Hword Block Read/Write subtype
	10:8	**Data Elements** Format: MDC_A64_DB_HW Specifies the number of contiguous Hwords to be read or written
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Hword Block Write MSD

MSD1W_A64_HWB - A64 Hword Block Write MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Block R/W
Group: HW Block R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 15h Format: Opcode A64 Hword Block Write message
	13	**Reserved**

MSD1W_A64_HWB - A64 Hword Block Write MSD

		Format: MBZ Ignored
	12:11	**A64 Block Message Subtype** Default Value: 3h Format: Opcode Hword Block Read/Write subtype
	10:8	**Data Elements** Format: **MDC_A64_DB_HW** Specifies the number of contiguous Hwords to be read or written
	7:0	**Binding Table Index** Format: **MDC_STATELESS** Specifies the message is stateless

A64 Oword Aligned Block Read MSD

MSD1R_A64_OWAB - A64 Oword Aligned Block Read MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Block R/W
Group: OW Aligned Block R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 14h Format: Opcode A64 Oword Block Read message
	13	**Reserved**

MSD1R_A64_OWAB - A64 Oword Aligned Block Read MSD

		Format: MBZ Ignored
	12:11	**A64 Block Message Subtype** Default Value: 1h Format: Opcode Oword Aligned Block Read subtype
	10:8	**Data Elements** Format: **MDC_A64_DB_OW** Specifies the number of contiguous Owords to be read
	7:0	**Binding Table Index** Format: **MDC_STATELESS** Specifies the message is stateless

A64 Oword Aligned Block Write MSD

MSD1W_A64_OWAB - A64 Oword Aligned Block Write MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Block R/W
Group: OW Aligned Block R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 15h Format: Opcode A64 Oword Block Write message
	13	**Reserved**

MSD1W_A64_OWAB - A64 Oword Aligned Block Write MSD

		Format: MBZ Ignored
	12:11	**A64 Block Message Subtype** Default Value: 1h Format: Opcode Oword Aligned Block Write subtype
	10:8	**Data Elements** Format: **MDC_A64_DB_OW** Specifies the number of contiguous Owords to be written
	7:0	**Binding Table Index** Format: **MDC_STATELESS** Specifies the message is stateless

A64 Oword Block Read MSD

MSD1R_A64_OWB - A64 Oword Block Read MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Block R/W
Group: OW Block R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 14h Format: Opcode A64 Oword Block Read message
	13	**Invalidate After Read**

MSD1R_A64_OWB - A64 Oword Block Read MSD

		Format: **MDC_IAR** Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12:11	**A64 Block Message Subtype** Default Value: 0h Format: Opcode Oword Block Read/Write subtype
	10:8	**Data Elements** Format: **MDC_A64_DB_OW** Specifies the number of contiguous Owords to be read or written
	7:0	**Binding Table Index** Format: **MDC_STATELESS** Specifies the message is stateless

A64 Oword Block Write MSD

MSD1W_A64_OWB - A64 Oword Block Write MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Block R/W
Group: OW Block R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 15h Format: Opcode A64 Oword Block Write message
	13	**Reserved**

MSD1W_A64_OWB - A64 Oword Block Write MSD

		Format: MBZ Ignored
	12:11	**A64 Block Message Subtype** Default Value: 0h Format: Opcode Oword Block Read/Write subtype
	10:8	**Data Elements** Format: **MDC_A64_DB_OW** Specifies the number of contiguous Owords to be read or written
	7:0	**Binding Table Index** Format: **MDC_STATELESS** Specifies the message is stateless

A64 Qword Scattered Read MSD

MSD1R_A64_QWS - A64 Qword Scattered Read MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Scattered R/W
Group: QW Scattered R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 10h Format: Opcode A64 Scattered Read message
	13	**Invalidate After Read**

MSD1R_A64_QWS - A64 Qword Scattered Read MSD

		Format: MDC_IAR Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12	**SIMD Mode** Format: MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)
	11:10	**Data Elements** Format: MDC_A64_DS Specifies the number of data elements to be read or written
	9:8	**A64 Scattered Message Subtype** Default Value: 2h Format: Opcode Qword Read/Write subtype
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Qword Scattered Write MSD

MSD1W_A64_QWS - A64 Qword Scattered Write MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Scattered R/W
Group: QW Scattered R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 1Ah Format: Opcode A64 Scattered Write message
	13	**Reserved**

MSD1W_A64_QWS - A64 Qword Scattered Write MSD

		Format: MBZ Ignored
	12	**SIMD Mode** Format: **MDC_SM2** Specifies the SIMD mode of the message (number of slots processed)
	11:10	**Data Elements** Format: **MDC_A64_DS** Specifies the number of data elements to be read or written
	9:8	**A64 Scattered Message Subtype** Default Value: 2h Format: Opcode Qword Read/Write subtype
	7:0	**Binding Table Index** Format: **MDC_STATELESS** Specifies the message is stateless

A64 Qword Untyped Atomic Integer Binary with Return Data Operation MSD

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

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Qword Untyped Atomic Integer Binary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF The message forbids a header
	18:14	**Message Type** Default Value: 12h Format: Opcode A64 Untyped Atomic Integer Operation message

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

	13	**Return Data Control** Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	**Data Width** Default Value: 1h Format: Opcode Operations are on 64-bit integers
	11:8	**Atomic Integer Operation** Format: MDC_AOP2 Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Qword Untyped Atomic Integer Binary Write Only Operation MSD

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

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Qword Untyped Atomic Integer Binary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF The message forbids a header
	18:14	**Message Type** Default Value: 12h Format: Opcode A64 Untyped Atomic Integer Operation message

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

	13	**Return Data Control** Default Value: 0h Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Data Width** Default Value: 1h Format: Opcode Operations are on 64-bit integers
	11:8	**Atomic Integer Operation** Format: MDC_AOP2 Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Qword Untyped Atomic Integer Ternary with Return Data Operation MSD

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

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Qword Untyped Atomic Integer Ternary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF The message forbids a header
	18:14	**Message Type** Default Value: 12h Format: Opcode A64 Untyped Atomic Integer Operation message

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

	13	**Return Data Control** Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	**Data Width** Default Value: 1h Format: Opcode Operations are on 64-bit integers
	11:8	**Atomic Integer Operation** Format: MDC_AOP3S Specifies the atomic integer operation to be performed. **Workaround** CMPWR_2W is not supported in A64 QWord SIMD8.
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Qword Untyped Atomic Integer Ternary Write Only Operation MSD

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

Source:	EuSubFunctionDataPort1
Length Bias:	1
Family:	Untyped Atomic Operation
Group:	Qword Untyped Atomic Integer Ternary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF The message forbids a header
	18:14	**Message Type** Default Value: 12h Format: Opcode A64 Untyped Atomic Integer Operation message

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

	13	**Return Data Control** Default Value: 0h Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Data Width** Default Value: 1h Format: Opcode Operations are on 64-bit integers
	11:8	**Atomic Integer Operation** Format: MDC_AOP3S Specifies the atomic integer operation to be performed. **Workaround** CMPWR_2W is not supported in A64 QWord SIMD8.
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Qword Untyped Atomic Integer Unary with Return Data Operation MSD

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

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Qword Untyped Atomic Integer Unary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF The message forbids a header
	18:14	**Message Type** Default Value: 12h Format: Opcode A64 Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.

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

	12	**Data Width** Default Value: 1h Format: Opcode Operations are on 64-bit integers
	11:8	**Atomic Integer Operation** Format: MDC_AOP1 Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Qword Untyped Atomic Integer Unary Write Only Operation MSD

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

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Qword Untyped Atomic Integer Unary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF The message forbids a header
	18:14	**Message Type** Default Value: 12h Format: Opcode A64 Untyped Atomic Integer Operation message

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

	13	**Return Data Control** Default Value: 0h Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Data Width** Default Value: 1h Format: Opcode Operations are on 64-bit integers
	11:8	**Atomic Integer Operation** Format: MDC_AOP1 Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Untyped Surface Read MSD

MSD1R_A64_US - A64 Untyped Surface Read MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Surface R/W
Group: Scattered Untyped Surface R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 11h Format: Opcode A64 Untyped Surface Read message
	13:12	**SIMD Mode**

MSD1R_A64_US - A64 Untyped Surface Read MSD

		Format: **MDC_SM3** Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Channel Mask** Format: **MDC_CMASK** Specifies which RGBA channels are included in the message payload.
	7:0	**Binding Table Index** Format: **MDC_STATELESS** Specifies the message is stateless

A64 Untyped Surface Write MSD

MSD1W_A64_US - A64 Untyped Surface Write MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Surface R/W
Group: Scattered Untyped Surface R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 19h Format: Opcode A64 Untyped Surface Write message
	13:12	**SIMD Mode** Format: MDC_SM3 Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Channel Mask** Format: MDC_UW_CMASK Specifies which RGBA channels are included in the message payload.
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

Addition

add - Addition

Source: Eulsa

Length Bias: 4

The add instruction performs component-wise addition of src0 and src1 and stores the results in dst. Addition of two floating-point numbers follows rules in add (IEEE mode) or add (ALT mode).

Format:

```
[(pred)] add[.cmod] (exec_size) dst src0 src1
```

Programming Notes

Use a source modifier with add to implement subtraction.

Syntax

```
[(pred)] add[.cmod] (exec_size) reg reg reg
         [(pred)] add[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
          for ( n = 0; n < exec_size; n++ ) {
             if ( WrEn.chan[n] ) {
                 dst.chan[n] = src0.chan[n] + src1.chan[n];
             }
          }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

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

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

add - Addition		
		Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: **EU_INSTRUCTION_SOURCES_REG_IMM**
	63:32	**Operand Controls** Format: **EU_INSTRUCTION_OPERAND_CONTROLS**
	31:0	**Header** Format: **EU_INSTRUCTION_HEADER**

Addition with Carry

addc - Addition with Carry

Source: Eulsa

Length Bias: 4

The addc instruction performs component-wise addition of src0 and src1 and stores the results in dst; it also stores the carry into acc. If the operation produces a carry out, 0x00000001 is stored in acc, else 0x00000000 is stored in acc.

Format:

```
[(pred)] addc[.cmod] (exec_size) dst src0 src1
```

Restriction

AccWrEn is required. The accumulator is an implicit destination and thus cannot be an explicit destination operand.

Syntax

```
[(pred)] addc[.cmod] (exec_size) reg reg reg
         [(pred)] addc[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
         for ( n = 0; n < exec_size; n++ ) {
           if ( WrEn.chan[n] ) {
             dst.chan[n] = src0.chan[n] + src1.chan[n];
             acc.chan[n] = carry(src0.chan[n] + src1.chan[n]);
           }
         }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	N	N

Src Types	Dst Types
UD	UD

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

addc - Addition with Carry			
	31:0	**Header**	
		Format:	**EU_INSTRUCTION_HEADER**

Arithmetic Shift Right

asr - Arithmetic Shift Right

Source: Eulsa

Length Bias: 4

Perform component-wise arithmetic right shift of the bits in src0 by the shift count indicated in src1, storing the results in dst. If src0 has a signed type, insert copies of src0's sign bit in the number of MSBs indicated by the shift count. Otherwise insert 0 bits. 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. In QWord mode, the shift count is taken from the low six bits of src1 regardless of the src1 type and treated as an unsigned integer in the range 0 to 63. Otherwise the shift count is taken from the low five bits of src1 regardless of the src1 type and treated as an unsigned integer in the range 0 to 31. The operation uses QWord mode if src0 or dst has the Q or UQ type but not if src1 is the only operand with the Q or UQ type. For positive values, this operation is src0 / 2shiftCount and for negative values, this operation is src0 / 2shiftCount - 1.

Format:

```
[(pred)] asr[.cmod] (exec_size) dst src0 src1
```

Programming Notes

If src0 is -1, the result is -1 regardless of the shift count.

For unsigned src0 types, asr and shr produce the same result.

Syntax

```
[(pred)] asr[.cmod] (exec_size) reg reg reg
         [(pred)] asr[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
          for ( n = 0; n < exec_size; n++ ) {
            if ( WrEn.channel[n] ) {
              shiftCnt = src1.chan[n] & 0x1F;  // Always use low 5 bits for shift count.
shiftCnt = src0 or dst has Q or UQ type ? src1.chan[n] & 0x3F : src1.chan[n] & 0x1F
              if (src0.chan[n] >= 0) {
                dst.chan[n] = src0.chan[n] » shiftCnt;
              }
              else {
                int maskLSB = pow(2, shiftCnt) - 1;
                if ( maskLSB & src0.chan[n] == 0 ) {
                  dst.chan[n] = sign(src0.chan[n]) * ((abs)src0.chan[n] » shiftCnt);
                }
                else {
                  dst.chan[n] = sign(src0.chan[n]) * ((abs)src0.chan[n] » shiftCnt) - 1;
                }
              }
            }
          }
```

asr - Arithmetic Shift Right

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

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

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

Average

avg - Average

Source: Eulsa

Length Bias: 4

The avg instruction performs component-wise integer average of src0 and src1 and stores the results in dst. An integer average uses integer upward rounding. It is equivalent to increment one to the addition of src0 and src1 and then apply an arithmetic right shift to this intermediate value.

Format:

```
       The avg instruction performs component-wise integer average of src0 and src1 and
stores the results in dst. An integer average uses integer upward rounding. It is
equivalent to increment one to the addition of src0 and src1 and then apply an arithmetic
right shift to this intermediate value.
```

Syntax

```
[(pred)] avg[.cmod] (exec_size) reg reg reg
         [(pred)] avg[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
          for ( n = 0; n < exec_size; n++ ) {
            if ( WrEn.chan[n] ) {
              dst.chan[n] = (src0.chan[n] + src1.chan[n] + 1) » 1; // Use arithmetic shift
right.
            }
          }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

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

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

avg - Average			
		Format:	EU_INSTRUCTION_HEADER

Barrier MSD

MSD_BARRIER - Barrier MSD

Source: EuSubFunctionGateway

Length Bias: 1

Record an additional thread reaching the barrier.

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Default Value: 0b Format: Enable **Restriction** Must be zero.
	18:3	**Reserved** Format: MBZ
	2:0	**Barrier Subfunction** Default Value: 100b Format: OpCode

Bit Field Extract

bfe - Bit Field Extract

Source: Eulsa

Length Bias: 4

Component-wise extract a bit field from src2 using the bit field width from src0 and the bit field offset from src1. Store the extracted bit field value in the low bits of dst and sign extend (if D type) or zero extend (if UD type). The width and offset values are from the low five bits of src0 and src1 respectively, or src0 & 0x1f and src1 & 0x1f. If width is zero, the result is zero. If offset + width > 32 then the extracted bit field is bits offset to 31 of src2, extracting only 32 - offset bits, less than width as the bit field cannot extend past the MSB of the source value. Otherwise extract width bits extending from bit positions offset to offset + width - 1.

Format:

```
[(pred)] bfe (exec_size) dst src0 src1 src2
```

Restriction

No accumulator access, implicit or explicit.

All three-source instructions have certain restrictions, described in Instruction Formats.

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
                   }
                }
             }
          }
```

bfe - Bit Field Extract

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

Src Types	Dst Types
UD	UD
D	D

DWord	Bit	Description
0..3	127	**Reserved** Format: MBZ
	126:106	**Source 2** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC
	105:85	**Source 1** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC
	84:64	**Source 0** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC
	63:56	**Destination Register Number** Format: DstRegNum
	55:53	**Destination Subregister Number**
	52:49	**Destination Channel Enable** Format: ChanEn[4] Four channel enables are defined for controlling which channels are written into the destination region. These channel mask bits are applied in a modulo-four manner to all ExecSize channels. There is 1-bit Channel Enable for each channel within the group of 4. If the bit is cleared, the write for the corresponding channel is disabled. If the bit is set, the write is enabled. Mnemonics for the bit being set for the group of 4 are "x", "y", "z", and "w", respectively, where "x" corresponds to Channel 0 in the group and "w" corresponds to channel 3 in the group
	48:46	**Destination Data Type** Value / Name / Description: 000b / :f / single precision Float (32-bit) 001b / :d / signed Doubleword integer 010b / :ud / Unsigned Doubleword integer 011b / :df / Double precision Float (64-bit) 100b / :hf / Half Float (16-bit)

bfe - Bit Field Extract

101b-111b	Reserved	

45:43 Source Data Type

Value	Name	Description
000b	:f	single precision Float (32-bit)
001b	:d	signed Doubleword integer
010b	:ud	Unsigned Doubleword integer
011b	:df	Double precision Float (64-bit)
100b	:hf	Half Float (16-bit)
101b-111b	Reserved	

42:41 Source 2 Modifier

Exists If:	(Property[Source Modifier]=='true')
Format:	**SrcMod**

42:37 Reserved

Exists If:	(Property[Source Modifier]=='false')
Format:	MBZ

40:39 Source 1 Modifier

Exists If:	(Property[Source Modifier]=='true')
Format:	**SrcMod**

38:37 Source 0 Modifier

Exists If:	(Property[Source Modifier]=='true')
Format:	**SrcMod**

36 Source 1 Type

Format:	U1

Only used if Source Data Type is :f or :hf, else Source 1 Data Type matches Source 0 type and this bit is ignored.

Value	Name	Description
0b	:f	single precision Float (32-bit)
1b	:hf	Half Float (16-bit)

35 Source 2 Type

Format:	U1

Only used if Source Data Type is :f or :hf, else Source 2 Data Type matches Source 0 type and this bit is ignored.

Value	Name	Description
0b	:f	single precision Float (32-bit)

bfe - Bit Field Extract

		1b :hf Half Float (16-bit)
	34	**MaskCtrl** (formerly WECtrl/Write Enable Control). This flag disables the normal write enables; it should normally be 0. **Value** / **Name** / **Description** 0 / Normal / Use the normal write enables in Dst.ChanEn (normal setting). 1 / NoMask / Write all channels except those disabled by predication or by other masks besides the write enables. **Programming Notes** MaskCtrl = NoMask also skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
	33	**Flag Register Number** This field contains the flag register number for instructions with a non-zero Conditional Modifier.
	32	**Flag Subregister Number** This field contains the flag subregister number for instructions with a non-zero Conditional Modifier.
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

Bit Field Insert 1

bfi1 - Bit Field Insert 1

Source: Eulsa

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

Bit Field Insert 2

bfi2 - Bit Field Insert 2

Source: Eulsa

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

No accumulator access, implicit or explicit.

All three-source instructions have certain restrictions, described in Instruction 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

bfi2 - Bit Field Insert 2

D	D

DWord	Bit	Description
0..3	127	**Reserved** Format: MBZ
	126:106	**Source 2** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC
	105:85	**Source 1** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC
	84:64	**Source 0** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC
	63:56	**Destination Register Number** Format: DstRegNum
	55:53	**Destination Subregister Number**
	52:49	**Destination Channel Enable** Format: ChanEn[4] Four channel enables are defined for controlling which channels are written into the destination region. These channel mask bits are applied in a modulo-four manner to all ExecSize channels. There is 1-bit Channel Enable for each channel within the group of 4. If the bit is cleared, the write for the corresponding channel is disabled. If the bit is set, the write is enabled. Mnemonics for the bit being set for the group of 4 are "x", "y", "z", and "w", respectively, where "x" corresponds to Channel 0 in the group and "w" corresponds to channel 3 in the group
	48:46	**Destination Data Type** Value / Name / Description: 000b / :f / single precision Float (32-bit) 001b / :d / signed Doubleword integer 010b / :ud / Unsigned Doubleword integer 011b / :df / Double precision Float (64-bit) 100b / :hf / Half Float (16-bit) 101b-111b / Reserved /
	45:43	**Source Data Type** Value / Name / Description: 000b / :f / single precision Float (32-bit)

001b	:d	signed Doubleword integer
010b	:ud	Unsigned Doubleword integer
011b	:df	Double precision Float (64-bit)
100b	:hf	Half Float (16-bit)
101b-111b	Reserved	

42:41 **Source 2 Modifier**

Exists If:	(Property[Source Modifier]=='true')
Format:	SrcMod

42:37 **Reserved**

Exists If:	(Property[Source Modifier]=='false')
Format:	MBZ

40:39 **Source 1 Modifier**

Exists If:	(Property[Source Modifier]=='true')
Format:	SrcMod

38:37 **Source 0 Modifier**

Exists If:	(Property[Source Modifier]=='true')
Format:	SrcMod

36 **Source 1 Type**

Format:	U1

Only used if Source Data Type is :f or :hf, else Source 1 Data Type matches Source 0 type and this bit is ignored.

Value	Name	Description
0b	:f	single precision Float (32-bit)
1b	:hf	Half Float (16-bit)

35 **Source 2 Type**

Format:	U1

Only used if Source Data Type is :f or :hf, else Source 2 Data Type matches Source 0 type and this bit is ignored.

Value	Name	Description
0b	:f	single precision Float (32-bit)
1b	:hf	Half Float (16-bit)

34 **MaskCtrl**

(formerly WECtrl/Write Enable Control). This flag disables the normal write enables; it should

bfi2 - Bit Field Insert 2

normally be 0.

Value	Name	Description
0	Normal	Use the normal write enables in Dst.ChanEn (normal setting).
1	NoMask	Write all channels except those disabled by predication or by other masks besides the write enables.

Programming Notes
MaskCtrl = NoMask also skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

33	**Flag Register Number** This field contains the flag register number for instructions with a non-zero Conditional Modifier.
32	**Flag Subregister Number** This field contains the flag subregister number for instructions with a non-zero Conditional Modifier.
31:0	**Header** Format: EU_INSTRUCTION_HEADER

Bit Field Reverse

bfrev - Bit Field Reverse

Source: Eulsa

Length Bias: 4

The bfrev instruction component-wise reverses all the bits in src0 and stores the results in dst.

Format:

```
[(pred)] bfrev (exec_size) dst src0
```

Restriction

No accumulator access, implicit or explicit.

Syntax

```
[(pred)] bfrev (exec_size) reg reg
[(pred)] bfrev (exec_size) reg imm32
```

Pseudocode

```
Evaluate(WrEn);
          for ( n = 0; n < exec_size; n++ ) {
             if ( WrEn.chan[n] ) {
                for ( idx = 0; idx < 32; idx++ ) {
                   dst.chan[n][idx] = src0.chan[n][31-idx];
                }
             }
          }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

Src Types	Dst Types
UD	UD

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

bfrev - Bit Field Reverse			
		Format:	EU_INSTRUCTION_HEADER

Branch Converging

brc - Branch Converging

Source: Eulsa

Length Bias: 4

Description

The brc instruction redirects the execution forward or backward to the instruction pointed by (current IP + offset). The jump will occur if all channels are branched away. UIP should reference the instruction where all channels are expected to come together. JIP should reference the end of the innermost conditional block.

In GEN binary, JIP and UIP use locations src1 and src0 respectively when immediate and location src0 when reg64, where reg64 is accessed as paired DWord (regioning being <2;2,1>). dst must be IP. When the offsets are immediate, src0 regfile must be immediate.

Format:

```
[(pred)] brc (exec_size) JIP UIP
```

Restriction

A brc instruction cannot use the Switch instruction option.

Syntax

```
[(pred)] brc (exec_size) imm32 imm32
[(pred)] brc (exec_size) reg64
```

Pseudocode

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

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

Src Types

D

DWord	Bit	Description
0..3	127:96	**JIP**

brc - Branch Converging

		Format: S31
		The byte-aligned jump distance if a jump is taken for the channel.
	95:64	**UIP**
		Format: S31
		The byte aligned jump distance if a jump is taken for the instruction.
	63:32	**Operand Control**
		Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header**
		Format: EU_INSTRUCTION_HEADER

Branch Diverging

brd - Branch Diverging

Source: Eulsa

Length Bias: 4

Description

The brd instruction redirects the execution forward or backward to the instruction pointed by (current IP + offset). The jump will occur if any channels are branched away.

In GEN binary, JIP is at location src1 when immediate and at location src0 when reg32, where reg32 is accessed as a scalar DWord. The ip register must be used (for example, by the assembler) as dst.

Format:

```
[(pred)] brd (exec_size) JIP
```

Restriction

A brd instruction cannot use the Switch instruction option.

Syntax

```
[(pred)] brd (exec_size) imm32
[(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	127:96	**JIP**

brd - Branch Diverging

		Format: S31 Jump Target Offset. The relative offset in bytes if a jump is taken for the instruction.
	95	**Source 0 Address Immediate [9] Sign Bit**
	94:91	**Src1.SrcType** Format: **SrcType**
	90:89	**Src1.RegFile** Format: **RegFile**
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align16') Format: **EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN16**
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1') Format: **EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN1**
	63:32	**Operand Control** Format: **EU_INSTRUCTION_OPERAND_CONTROLS**
	31:0	**Header** Format: **EU_INSTRUCTION_HEADER**

Break

break - Break

Source: Eulsa

Length Bias: 4

Description

The break instruction is used to early-out from the inner most loop, or early out from the inner most switch block. When used in a loop, upon execution, the break instruction terminates the loop for all execution channels enabled. If all the enabled channels hit the break instruction, jump to the instruction referenced by JIP. JIP should be the offset to the end of the inner most conditional or loop block, UIP should be the offset to the while instruction of the loop block. If SPF is ON, the UIP must be used to update IP; JIP is not used in this case

The following table describes the two 32-bit instruction pointer offsets. Both the JIP and UIP are signed 32-bit numbers, added to IP pre-increment. In GEN binary, JIP and UIP are at locations src0 and src1 and must be of type DW (signed DWord integer). When the offsets are immediate, src0 regfile must be immediate.

Format:

```
[(pred)] break (exec_size) JIP UIP
```

Syntax

```
[(pred)] break (exec_size) imm32 imm32
```

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	127:96	**JIP** Format: S31 The byte-aligned jump distance if a jump is taken for the channel.

break - Break		
	95:64	**UIP** Format: S31 The byte aligned jump distance if a jump is taken for the instruction.
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

Byte Scaled Read MSD

MSD2R_BS - Byte Scaled Read MSD

Source: EuSubFunctionDataPort2
Length Bias: 1
Family: Scaled R/W
Group: Byte Scaled R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_A32_MHP If present, modifies the address calculations.
	18:15	**Message Type** Default Value: 04h Format: Opcode Byte Scattered Read message
	14:12	**Reserved** Format: MBZ Ignored
	11:10	**Data Elements** Format: MDC_DS Specifies the number of Bytes to be read or written per Dword
	9	**Reserved** Format: MBZ Ignored
	8	**SIMD Mode** Format: MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)

MSD2R_BS - Byte Scaled Read MSD		
	7:0	**Sideband Scaled Offset** Format: MDC_A32_SBSO In combination with Header Present field, specifies the Scale pitch and the Offset for the message.

Byte Scaled Write MSD

MSD2W_BS - Byte Scaled Write MSD

Source: EuSubFunctionDataPort2
Length Bias: 1
Family: Scaled R/W
Group: Byte Scaled R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_A32_MHP If present, modifies the address calculations.
	18:15	**Message Type** Default Value: 0Ch Format: Opcode Byte Scattered Write message
	14:12	**Reserved**

MSD2W_BS - Byte Scaled Write MSD

		Format: MBZ Ignored
	11:10	**Data Elements** Format: **MDC_DS** Specifies the number of Bytes to be read or written per Dword
	9	**Reserved** Format: MBZ Ignored
	8	**SIMD Mode** Format: **MDC_SM2** Specifies the SIMD mode of the message (number of slots processed)
	7:0	**Sideband Scaled Offset** Format: **MDC_A32_SBSO** In combination with Header Present field, specifies the Scale pitch and the Offset for the message.

Byte Scattered Read MSD

MSD0R_BS - Byte Scattered Read MSD

Source: EuSubFunctionDataPort0
Length Bias: 1
Family: Scattered R/W
Group: Byte Scattered R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: Enable If set, indicates that the message includes the header.
	18	**Legacy Message** Default Value: 0h Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 04h

MSD0R_BS - Byte Scattered Read MSD

		Format: Opcode Byte Scattered Read message
	13	**Reserved** Format: MBZ Ignored
	12	**Reserved** Format: MBZ Ignored
	11:10	**Data Elements** Format: MDC_DS Specifies the number of Bytes to be read or written per Dword
	9	**Reserved** Format: MBZ Ignored
	8	**SIMD Mode** Format: MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies the Binding Table Index for the message

Byte Scattered Write MSD

MSD0W_BS - Byte Scattered Write MSD

Source: EuSubFunctionDataPort0
Length Bias: 1
Family: Scattered R/W
Group: Byte Scattered R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: Enable If set, indicates that the message includes the header.
	18	**Legacy Message** Default Value: 0h Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 0Ch

MSD0W_BS - Byte Scattered Write MSD

		Format: Opcode Byte Scattered Write message
	13:12	**Reserved** Format: MBZ Ignored
	11:10	**Data Elements** Format: MDC_DS Specifies the number of Bytes to be read or written per Dword
	9	**Reserved** Format: MBZ Ignored
	8	**SIMD Mode** Format: MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies the Binding Table Index for the message

Call

call - Call

Source: Eulsa

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 section describes JIP, the jump offset.
JIP can be an immediate or register value. When a jump occurs, this value is added to IP pre-increment. In GEN binary, JIP is at location src1 and src0 must be null. The GRF register must be put (for example, by the assembler) at dst location.
Format: [(pred)] call (exec_size) dst JIP

Format:

```
[(pred)] call (exec_size) dst JIP
```

Restriction

The call instruction must have DWord source and destination type, and the destination must be QWord aligned.

A call instruction must target an instruction with Switch set; in addition, the instruction following the call must also have Switch set.

Syntax

```
[(pred)] call (exec_size) reg imm32 [(pred)] call (exec_size) reg reg32
```

Pseudocode

```
Evaluate(WrEn);
          for ( n = 0; n < exec_size; n++ ) {
             if (WrEn.chan[n] ) {
                PcIP[n] = IP + JIP;
                CallMask[n] = 1;
             }
             else {
                PcIP[n] = IP + 1;
                CallMask[n] = 0;
             }
          }

          if ( PcIP[n] != (IP + 1) ) {  // any channel jumped
             dst.chan[0] = IP + 1;
```

call - Call

```
            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	127:96	**JIP** Format: S31 Jump Target Offset. The relative offset in bytes if a jump is taken for the instruction.
	95	**Source 0 Address Immediate [9] Sign Bit**
	94:91	**Src1.SrcType** Format: **SrcType**
	90:89	**Src1.RegFile** Format: **RegFile**
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align16') Format: **EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN16**
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1') Format: **EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN1**
	63:32	**Operand Control** Format: **EU_INSTRUCTION_OPERAND_CONTROLS**
	31:0	**Header** Format: **EU_INSTRUCTION_HEADER**

Call Absolute

calla - Call Absolute

Source: Eulsa

Length Bias: 4

The calla instruction jumps to a subroutine. It can be predicated or non-predicated. If non-predicated, all enabled channels jump to the subroutine. If predicated, only the channels enabled by PMask jump to the subroutine; the rest of the channels move to the next instruction after the calla instruction. If none of the channels jump into the subroutine, the calla instruction is treated as a nop. In case of a jump, the call instruction stores the return IP onto the first DWord of the destination register and stores the CallMask in the second DWord of the destination register. If SPF is ON, none of the PcIP are updated. When SPF is on, the predication control must be scalar. The difference between calla and call is that calla uses JIP as the IP value rather than adding it to the IP value.

Format:

```
[(pred)] calla (exec_size) dst JIP
```

Restriction

The calla instruction must have DWord source and destination type, and the destination must be QWord-aligned.

Syntax

```
[(pred)] calla (exec_size) reg imm32
```

Pseudocode

```
Evaluate(WrEn);
          for ( n = 0; n < exec_size; n++ ) {
             if ( WrEn.channel[n] ) {
                PcIP[n] = JIP;
                CallMask[n] = 1;
             }
             else {
                PcIP[n] = IP + 1;
                CallMask[n] = 0;
             }
          }

          if ( PcIP[n] != (IP + 1) ) {  // any channel jumped
             dst.chan[0] = IP + 1;
             dst.chan[1] = CallMask;
             Jump(JIP);
          }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

calla - Call Absolute

Dst Types
D, UD

DWord	Bit	Description
0..3	127:96	**JIP** Format: S31 Jump Target Offset. The relative offset in bytes if a jump is taken for the instruction.
	95	**Source 0 Address Immediate [9] Sign Bit**
	94:91	**Src1.SrcType** Format: SrcType
	90:89	**Src1.RegFile** Format: RegFile
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align16') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN16
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN1
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

Compare

cmp - Compare

Source: Eulsa

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
```

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

cmp - Compare

HF	HF
*B,*W,*D	HF
*W,*D	*W,*D
HF, F	HF, F

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

Compare NaN

cmpn - Compare NaN

Source: Eulsa

Length Bias: 4

The cmpn instruction performs component-wise special-NaN comparison of src0 and src1 and stores the results in the selected flag register and in dst. It takes component-wise subtraction of src0 and src1, evaluating the conditional signals including NS based on the conditional modifier, and storing the conditional flag bits in bit-packed form in the destination flag register and all bits of dst channels. If the dst is not null, for the enabled channels, then all bits of the destination channel will contain the flag value for the channel. When the instruction operates on packed word format, one general register may store up to 16 such comparison results. In DWord format, one general register may store up to 8 results. A conditional modifier must be specified; the conditional modifier field cannot be 0000b. More information about the conditional signals used is in the Creating Conditional Flags section. For each enabled channel 0b or 1b is assigned to the appropriate flag bit and 0/all zeros or all ones (e.g, byte 0xFF, word 0xFFFF, DWord 0xFFFFFFFF) is assigned to dst. Min/Max instructions use cmpn to select the destination from the input sources (see the Min Max of Floating Point Numbers section for details).

Format:

```
[(pred)] cmpn[.cmod] (exec_size) dst src0 src1
```

Restriction

.l and .ge are the only two conditional modifiers are supported for this instruction.

Syntax

```
[(pred)] cmpn[.cmod] (exec_size) reg reg reg
[(pred)] cmpn[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
       for ( n = 0; n < exec_size; n++ ) {
         bitMask[n] = 0;
         if ( WrEn.chan[n] ) {
           results[n] = src0.chan[n] - src1.chan[n];
           bitMask[n] = ConditionNaN(results[n]);
           dst.chan[n][0] = bitMask[n];  // All bits for dst channel
       }
     }
     flag# = bitMask;
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	N	Y

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

cmpn - Compare NaN

*B,*W,*D	F
F	F
HF	HF
F, HF	F, HF

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

Conditional Select

csel - Conditional Select

Source: Eulsa

Length Bias: 4

Description

The csel instruction selectively moves components in src0 or src1 to the dst based on the result of the compare of src2 with zero. If the channel condition is true, data in src0 is moved into dst. Otherwise, data in src1 is moved into dst. The csel instruction provides the function of a cmp followed by sel. The instruction must not be used if cmpn is required. The instruction does not update the flag register.
The comparison follows the same rule as cmp instruction for that data type.

When Access Mode is Align1, accumulator may be used as source or destination.

Format:

```
csel (exec_size) dst src0 src1 src2
```

Syntax

csel[.cmod] (exec_size) reg reg reg reg

Pseudocode

```
Evaluate(WrEn);
          for ( n = 0; n < exec_size; n++ ) {
              bitMask[n] = 0;
              if ( EMask.chan[n] ) {
                 result[n] = src2.chan[n] - 0;
                 bitMask[n] = Condition(result[n]);
                 if (bitMask[n] = 1) {
                    dst.chan[n] = src0.chan[n];
                 }
                 else {
                    dst.chan[n] = src1.chan[n];
                 }
              }
           }
```

Predication	Conditional Modifier	Saturation	Source Modifier
N	Y	Y	Y

Src Types	Dst Types
F	F
HF	HF
D	D
W	W

DWord	Bit	Description

csel - Conditional Select

0..3	127	**Reserved** Format: MBZ
	126:106	**Source 2** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC
	105:85	**Source 1** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC
	84:64	**Source 0** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC
	63:56	**Destination Register Number** Format: DstRegNum
	55:53	**Destination Subregister Number**
	52:49	**Destination Channel Enable** Format: ChanEn[4] Four channel enables are defined for controlling which channels are written into the destination region. These channel mask bits are applied in a modulo-four manner to all ExecSize channels. There is 1-bit Channel Enable for each channel within the group of 4. If the bit is cleared, the write for the corresponding channel is disabled. If the bit is set, the write is enabled. Mnemonics for the bit being set for the group of 4 are "x", "y", "z", and "w", respectively, where "x" corresponds to Channel 0 in the group and "w" corresponds to channel 3 in the group
	48:46	**Destination Data Type** Value / Name / Description 000b / :f / single precision Float (32-bit) 001b / :d / signed Doubleword integer 010b / :ud / Unsigned Doubleword integer 011b / :df / Double precision Float (64-bit) 100b / :hf / Half Float (16-bit) 101b-111b / Reserved /
	45:43	**Source Data Type** Value / Name / Description 000b / :f / single precision Float (32-bit) 001b / :d / signed Doubleword integer 010b / :ud / Unsigned Doubleword integer

011b	:df	Double precision Float (64-bit)
100b	:hf	Half Float (16-bit)
101b-111b	Reserved	

42:41 Source 2 Modifier

Exists If:	(Property[Source Modifier]=='true')
Format:	SrcMod

42:37 Reserved

Exists If:	(Property[Source Modifier]=='false')
Format:	MBZ

40:39 Source 1 Modifier

Exists If:	(Property[Source Modifier]=='true')
Format:	SrcMod

38:37 Source 0 Modifier

Exists If:	(Property[Source Modifier]=='true')
Format:	SrcMod

36 Source 1 Type

Format:	U1

Only used if Source Data Type is :f or :hf, else Source 1 Data Type matches Source 0 type and this bit is ignored.

Value	Name	Description
0b	:f	single precision Float (32-bit)
1b	:hf	Half Float (16-bit)

35 Source 2 Type

Format:	U1

Only used if Source Data Type is :f or :hf, else Source 2 Data Type matches Source 0 type and this bit is ignored.

Value	Name	Description
0b	:f	single precision Float (32-bit)
1b	:hf	Half Float (16-bit)

34 MaskCtrl

(formerly WECtrl/Write Enable Control). This flag disables the normal write enables; it should normally be 0.

Value	Name	Description

		0 Normal Use the normal write enables in Dst.ChanEn (normal setting). 1 NoMask Write all channels except those disabled by predication or by other masks besides the write enables. **Programming Notes** MaskCtrl = NoMask also skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
	33	**Flag Register Number** This field contains the flag register number for instructions with a non-zero Conditional Modifier.
	32	**Flag Subregister Number** This field contains the flag subregister number for instructions with a non-zero Conditional Modifier.
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

Conditional Send Message

sendc - Conditional Send Message

Source: Eulsa

Length Bias: 4

The sendc instruction has the same behavior as the send instruction except the following. sendc first checks the dependent threads inside the Thread Dependency Register. There are up to 8 dependent threads in the TDR register. The sendc instruction executes only when all the dependent threads in the TDR register are retired. Wait for dependencies in the TDR Register to clear, then send a message stored in registers starting at src to a shared function identified by exdesc along with control from desc with a general register writeback location at dst.

Format:

```
[(pred)] sendc (exec_size) dst src0 exdesc desc
```

Restriction

The sendc instruction has the same restrictions as the send instruction.

Syntax

```
[(pred)] sendc (exec_size) reg reg imm32 reg32a
              [(pred)] sendc (exec_size) reg reg imm32 imm32
```

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	**Reserved** Format: MBZ
	94:91	**ExDesc[31:28]** Format: ExtMsgDescpt[31:28]
	90:89	**Reserved** Format: MBZ

sendc - Conditional Send Message

	88:85	**ExDesc[27:24]** Format: **ExtMsgDescpt[27:24]**
	84	**Reserved** Format: MBZ
	83:80	**ExDesc[23:20]** Format: **ExtMsgDescpt[23:20]**
	79:68	**Reserved** Format: MBZ
	67:64	**ExDesc[19:16]** Format: ExMsgDescpt[19:16]
	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**

Conditional Split Send Message

sendsc - Conditional Split Send Message

Source: Eulsa

Length Bias: 4

The sendsc instruction has the same behavior as the sends instruction except the following. sendsc first checks the dependent threads inside the Thread Dependency Register. There are up to 8 dependent threads in the TDR register. The sendsc 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 GRF locations starting at <src0> followed by <src1> to a shared function identified by <ex_desc> along with control from <desc> and <ex_desc> with a GRF writeback location at <dest>.

Format:

```
[(pred)] sendsc (exec_size) <dest> <src0> <src1> <ex_desc> <desc>
```

Restriction

The sendsc instruction has the same restrictions as the sends instruction.

Syntax

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

Pseudocode

if (TDR[7] ... || TDR[2] || TDR[1] || TDR[0]) { wait; } Evaluate(WrEn); <MsgChEnable> = WrEn; <SourceReg0> = <src0>.RegNum; <SourceReg1> = <src1>.RegNum; MessageEnqueue(<MsgChEnable>, <ResponseReg>, <SourceReg0>, <SourceReg1>,<ex_desc>, <dest>);

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

DWord	Bit	Description
0..3	127:96	**Message** Format: EU_INSTRUCTION_OPERAND_SEND_MSG
	95:80	**ExDesc[31:16]** Format: ExtMsgDescpt[31:16]
	79	**Source 0 Addressing Mode** Format: AddrMode
	78	**Reserved**

		Exists If: ([Source 0 Addressing Mode]=='Direct') Format: MBZ
	78	**Source 0 Address Immediate Sign [9]** Exists If: ([Source 0 Addressing Mode]=='Indirect') Format: S9[9]
	77	**SelReg32Desc** Indicate the source of Message Descriptor. Immediate value from instruction of indirect value from address registers. Value: 0 — IMM Value: 1 — REG32
	76:73	**Source 0 Address Subregister Number** Exists If: ([Source 0 Addressing Mode]=='Indirect')
	76:69	**Source 0 Register Number** Exists If: ([Source 0 Addressing Mode]=='Direct')
	72:68	**Source 0 Address Immediate [8:4]** Exists If: ([Source 0 Addressing Mode]=='Indirect') Format: S9[8:4]
	68	**Source 0 Subregister Number** Exists If: ([Source 0 Addressing Mode]=='Direct')
	67:64	**ExDesc[9:6]** Format: ExtMsgDescpt[9:6]
	63	**Destination Addressing Mode** Format: AddrMode
	62	**Destination Address Immediate Sign [9]** Exists If: ([Destination Addressing Mode]=='Indirect') Format: S9[9]
	62	**Reserved** Exists If: ([Destination Addressing Mode]=='Direct') Format: MBZ
	61	**SelReg32ExDesc** Indicate the source of Extended Message Descriptor. Immediate value from instruction of indirect value from address registers. Value: 0 — IMM Value: 1 — REG32

sendsc - Conditional Split Send Message

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

Constant Cache Dword Scattered Read MSD

MSD_CC_DWS - Constant Cache Dword Scattered Read MSD

Source: EuSubFunctionReadOnlyDataPort
Length Bias: 1
Family: Scattered R/W
Group: DW Scattered R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: Enable If set, indicates that the message includes the header.
	18	**Legacy Message** Default Value: 0h Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 03h

MSD_CC_DWS - Constant Cache Dword Scattered Read MSD

		Format: Opcode Dword Scattered Read message
	13	**Invalidate After Read** Format: MDC_IAR Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12	**Reserved** Format: MBZ Ignored
	11:10	**Reserved** Format: MBZ Ignored
	9	**Legacy SIMD Mode** Default Value: 1h Format: Opcode Must be set for compatibility.
	8	**SIMD Mode** Format: MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Constant Cache Oword Aligned Block Read MSD

MSD_CC_OWAB - Constant Cache Oword Aligned Block Read MSD

Source: EuSubFunctionReadOnlyDataPort
Length Bias: 1
Family: Block R/W
Group: OW Aligned Block R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18	**Reserved** Format: MBZ Ignored
	17:14	**Message Type** Default Value: 01h

MSD_CC_OWAB - Constant Cache Oword Aligned Block Read MSD

		Format: Opcode Oword Aligned Block Read Constant Cache message
	13:11	**Reserved** Format: MBZ Ignored
	10:8	**Data Elements** Format: MDC_DB_OW Specifies the number of contiguous Owords to be read
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Constant Cache Oword Block Read MSD

MSD_CC_OWB - Constant Cache Oword Block Read MSD

Source: EuSubFunctionReadOnlyDataPort
Length Bias: 1
Family: Block R/W
Group: OW Block R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18	**Reserved** Format: MBZ Ignored
	17:14	**Message Type** Default Value: 00h

MSD_CC_OWB - Constant Cache Oword Block Read MSD

		Format: Opcode Oword Block Read Constant Cache message
	13	**Invalidate After Read** Format: MDC_IAR Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12:11	**Reserved** Format: MBZ Ignored
	10:8	**Data Elements** Format: MDC_DB_OW Specifies the number of contiguous Owords to be read or written
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Continue

cont - Continue

Source: Eulsa

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 instruction or right before the condition check code block for the while instruction. If all enabled channels hit this instruction, jump to the instruction referenced by JIP where execution continues. UIP should always reference the loop's associated while instruction. JIP should point to the last instruction of the inner most conditional block if the cont instruction is inside a conditional block. In case of the break instruction directly under the loop, the JIP and the UIP are the same. If SPF is ON, the UIP must be used to update IP; JIP is not used in this case.

The following table describes the two 32-bit instruction pointer offsets. Both the JIP and UIP are signed 32-bit numbers, added to IP pre-increment. In GEN binary, JIP and UIP are at locations src0 and src1 and must be of type DW (signed DWord integer). When the offsets are immediate, src0 regfile must be immediate.

Format:

```
[(pred)] cont (exec_size) JIP UIP
```

Restriction

The execution size must be the same for the while, break, and cont instructions of the same code block.

Syntax

```
[(pred)] cont (exec_size) imm32 imm32
```

Pseudocode

```
Evaluate(WrEn);
          for ( n = 0; n < exec_size; n++ ) {
             if ( WrEn.channel[n] ) {
                if ( PMask[n] ) {  // PMask is for all channels enabled for the cont
instruction.
                   PcIP[n] = IP + UIP;
                }
                else {
                   PcIP[n] = IP + 1;
                }
             }
          }
          for ( n = exec_size; n < 32; n++ ) {
             PcIP[n] = IP + 1;
          }
          if ( PcIP != (IP + 1) ) {  // all channels true
             Jump(IP + JIP);
          }
```

cont - Continue

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

DWord	Bit	Description
0..3	127:96	**JIP** Format: S31 The byte-aligned jump distance if a jump is taken for the channel.
	95:64	**UIP** Format: S31 The byte aligned jump distance if a jump is taken for the instruction.
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

Count Bits Set

cbit - Count Bits Set

Source: Eulsa

Length Bias: 4

The cbit instruction counts component-wise the total bits set in src0 and stores the resulting counts in dst.

Format:

```
[(pred)] cbit (exec_size) dst src0
```

Restriction

No accumulator access, implicit or explicit.

Syntax

```
[(pred)] cbit (exec_size) reg reg
[(pred)] cbit (exec_size) reg imm32
```

Pseudocode

```
Evaluate(WrEn);
for ( n = 0; n < exec_size; n++ ) {
   if ( WrEn.chan[n] ) {
       UD cnt = 0;
       UD val = src0.chan[n];
       while ( val ) {
         if ( val & 1 ) {
           cnt ++;
         }
         val = val » 1;
       }
       dst.chan[n] = cnt;
   }
}
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

Src Types	Dst Types
UB, UW, UD	UD

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

cbit - Count Bits Set		
		Format: EU_INSTRUCTION_SOURCES_IMM32
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

Dword Atomic Counter Binary with Return Data Operation MSD

MSD1R_DWAC2 - Dword Atomic Counter Binary with Return Data Operation MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Dword Atomic Counter Binary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header
	18:14	**Message Type** Default Value: 0Bh Format: Opcode Atomic Counter Operation message
	13	**Return Data Control** Default Value: 1h

MSD1R_DWAC2 - Dword Atomic Counter Binary with Return Data Operation MSD

		Format: Opcode Specifies that return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2RS Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Atomic Integer Operation** Format: MDC_AOP2 Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Dword Atomic Counter Binary Write Only Operation MSD

MSD1W_DWAC2 - Dword Atomic Counter Binary Write Only Operation MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Dword Atomic Counter Binary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header
	18:14	**Message Type** Default Value: 0Bh Format: Opcode Atomic Counter Operation message
	13	**Return Data Control** Default Value: 0h

MSD1W_DWAC2 - Dword Atomic Counter Binary Write Only Operation MSD

		Format: Opcode
		Specifies that no return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2RS Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Atomic Integer Operation** Format: MDC_AOP2 Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Dword Atomic Counter Unary with Return Data Operation MSD

MSD1R_DWAC1 - Dword Atomic Counter Unary with Return Data Operation MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Dword Atomic Counter Unary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header
	18:14	**Message Type** Default Value: 0Bh Format: Opcode Atomic Counter Operation message
	13	**Return Data Control** Default Value: 1h

MSD1R_DWAC1 - Dword Atomic Counter Unary with Return Data Operation MSD

		Format: Opcode Specifies that return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2RS Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Atomic Integer Operation** Format: MDC_AOP1 Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Dword Atomic Counter Unary Write Only Operation MSD

MSD1W_DWAC1 - Dword Atomic Counter Unary Write Only Operation MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Dword Atomic Counter Unary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header
	18:14	**Message Type** Default Value: 0Bh Format: Opcode Atomic Counter Operation message
	13	**Return Data Control** Default Value: 0h

MSD1W_DWAC1 - Dword Atomic Counter Unary Write Only Operation MSD

		Format: Opcode Specifies that no return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2RS Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Atomic Integer Operation** Format: MDC_AOP1 Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Dword Scattered Read MSD

MSD0R_DWS - Dword Scattered Read MSD

Source: EuSubFunctionDataPort0
Length Bias: 1
Family: Scattered R/W
Group: DW Scattered R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: Enable If set, indicates that the message includes the header.
	18	**Legacy Message** Default Value: 0h Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 03h

MSD0R_DWS - Dword Scattered Read MSD

		Format: Opcode Dword Scattered Read message
	13	**Invalidate After Read** Format: MDC_IAR Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12	**Reserved** Format: MBZ Ignored
	11:10	**Reserved** Format: MBZ Ignored
	9	**Legacy SIMD Mode** Default Value: 1h Format: Opcode Must be set for compatibility.
	8	**SIMD Mode** Format: MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)
	7:0	**Binding Table Index** Format: MDC_BTS_A32 Specifies the Binding Table Index for the message

Dword Scattered Write MSD

MSD0W_DWS - Dword Scattered Write MSD

Source: EuSubFunctionDataPort0
Length Bias: 1
Family: Scattered R/W
Group: DW Scattered R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: Enable If set, indicates that the message includes the header.
	18	**Legacy Message** Default Value: 0h Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 0Bh

MSD0W_DWS - Dword Scattered Write MSD

		Format: Opcode Dword Scattered Write message
	13:12	**Reserved** Format: MBZ Ignored
	11:10	**Reserved** Format: MBZ Ignored
	9	**Legacy SIMD Mode** Default Value: 1h Format: Opcode Must be set for compatibility.
	8	**SIMD Mode** Format: MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)
	7:0	**Binding Table Index** Format: MDC_BTS_A32 Specifies the Binding Table Index for the message

Dword Typed Atomic Integer Binary with Return Data Operation MSD

MSD1R_DWTAI2 - Dword Typed Atomic Integer Binary with Return Data Operation MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Typed Atomic Operation
Group: Dword Typed Atomic Integer Binary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 06h Format: Opcode Typed Atomic Integer Operation message
	13	**Return Data Control**

MSD1R_DWTAI2 - Dword Typed Atomic Integer Binary with Return Data Operation MSD

		Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	**Slot Group** Format: MDC_SG2 Specifies the Slot Group mode of the message (which slots are processed)
	11:8	**Atomic Integer Operation** Format: MDC_AOP2 Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Dword Typed Atomic Integer Binary Write Only Operation MSD

MSD1W_DWTAI2 - Dword Typed Atomic Integer Binary Write Only Operation MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Typed Atomic Operation
Group: Dword Typed Atomic Integer Binary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 06h Format: Opcode Typed Atomic Integer Operation message
	13	**Return Data Control** Default Value: 0h

MSD1W_DWTAI2 - Dword Typed Atomic Integer Binary Write Only Operation MSD

		Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Slot Group** Format: MDC_SG2 Specifies the Slot Group mode of the message (which slots are processed)
	11:8	**Atomic Integer Operation** Format: MDC_AOP2 Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Dword Typed Atomic Integer Ternary with Return Data Operation MSD

MSD1R_DWTAI3 - Dword Typed Atomic Integer Ternary with Return Data Operation MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Typed Atomic Operation
Group: Dword Typed Atomic Integer Ternary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 06h Format: Opcode Typed Atomic Integer Operation message
	13	**Return Data Control**

MSD1R_DWTAI3 - Dword Typed Atomic Integer Ternary with Return Data Operation MSD

		Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	**Slot Group** Format: MDC_SG2 Specifies the Slot Group mode of the message (which slots are processed)
	11:8	**Atomic Integer Operation** Format: MDC_AOP3S Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Dword Typed Atomic Integer Ternary Write Only Operation MSD

MSD1W_DWTAI3 - Dword Typed Atomic Integer Ternary Write Only Operation MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Typed Atomic Operation
Group: Dword Typed Atomic Integer Ternary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 06h Format: Opcode Typed Atomic Integer Operation message
	13	**Return Data Control** Default Value: 0h

MSD1W_DWTAI3 - Dword Typed Atomic Integer Ternary Write Only Operation MSD

		Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Slot Group** Format: MDC_SG2 Specifies the Slot Group mode of the message (which slots are processed)
	11:8	**Atomic Integer Operation** Format: MDC_AOP3S Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Dword Typed Atomic Integer Unary with Return Data Operation MSD

MSD1R_DWTAI1 - Dword Typed Atomic Integer Unary with Return Data Operation MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Typed Atomic Operation
Group: Dword Typed Atomic Integer Unary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 06h Format: Opcode Typed Atomic Integer Operation message
	13	**Return Data Control**

MSD1R_DWTAI1 - Dword Typed Atomic Integer Unary with Return Data Operation MSD

		Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	**Slot Group** Format: MDC_SG2 Specifies the Slot Group mode of the message (which slots are processed)
	11:8	**Atomic Integer Operation** Format: MDC_AOP1 Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Dword Typed Atomic Integer Unary Write Only Operation MSD

MSD1W_DWTAI1 - Dword Typed Atomic Integer Unary Write Only Operation MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Typed Atomic Operation
Group: Dword Typed Atomic Integer Unary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 06h Format: Opcode Typed Atomic Integer Operation message
	13	**Return Data Control** Default Value: 0h

MSD1W_DWTAI1 - Dword Typed Atomic Integer Unary Write Only Operation MSD

		Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Slot Group** Format: MDC_SG2 Specifies the Slot Group mode of the message (which slots are processed)
	11:8	**Atomic Integer Operation** Format: MDC_AOP1 Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Dword Untyped Atomic Float Binary with Return Data Operation MSD

MSD1R_DWAF2 - Dword Untyped Atomic Float Binary with Return Data Operation MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Dword Untyped Atomic Float Binary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 1Bh Format: Opcode Untyped Atomic Float Operation message
	13	**Return Data Control**

MSD1R_DWAF2 - Dword Untyped Atomic Float Binary with Return Data Operation MSD

		Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2R Specifies the SIMD mode of the message (number of slots processed)
	11	**Data Width** Default Value: 0h Format: Opcode Operations are on 32-bit floats.
	10:8	**Atomic Float Operation** Format: MDC_FOP2 Specifies the atomic float operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies the Binding Table Index for the message

Dword Untyped Atomic Float Binary Write Only Operation MSD

MSD1W_DWAF2 - Dword Untyped Atomic Float Binary Write Only Operation MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Dword Untyped Atomic Float Binary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 1Bh Format: Opcode Untyped Atomic Float Operation message
	13	**Return Data Control** Default Value: 0h

MSD1W_DWAF2 - Dword Untyped Atomic Float Binary Write Only Operation MSD

		Format: Opcode
		Specifies that no return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2R Specifies the SIMD mode of the message (number of slots processed)
	11	**Data Width** Default Value: 0h Format: Opcode Operations are on 32-bit floats.
	10:8	**Atomic Float Operation** Format: MDC_FOP2 Specifies the atomic float operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies the Binding Table Index for the message

Dword Untyped Atomic Float Ternary with Return Data Operation MSD

MSD1R_DWAF3 - Dword Untyped Atomic Float Ternary with Return Data Operation MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Dword Untyped Atomic Float Ternary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 1Bh Format: Opcode Untyped Atomic Float Operation message
	13	**Return Data Control**

MSD1R_DWAF3 - Dword Untyped Atomic Float Ternary with Return Data Operation MSD

		Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2R Only SIMD8 operations are supported.
	11	**Data Width** Default Value: 0h Format: Opcode Operations are on 32-bit floats.
	10:8	**Atomic Float Operation** Format: MDC_FOP3 Specifies the atomic float operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies the Binding Table Index for the message

Dword Untyped Atomic Float Ternary Write Only Operation MSD

MSD1W_DWAF3 - Dword Untyped Atomic Float Ternary Write Only Operation MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Dword Untyped Atomic Float Ternary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 1Bh Format: Opcode Untyped Atomic Float Operation message
	13	**Return Data Control** Default Value: 0h

MSD1W_DWAF3 - Dword Untyped Atomic Float Ternary Write Only Operation MSD

		Format: Opcode
		Specifies that no return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2R Specifies the SIMD mode of the message (number of slots processed)
	11	**Data Width** Default Value: 0h Format: Opcode Operations are on 32-bit floats.
	10:8	**Atomic Float Operation** Format: MDC_FOP3 Specifies the atomic float operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies the Binding Table Index for the message

Dword Untyped Atomic Integer Binary with Return Data Operation MSD

MSD1R_DWAI2 - Dword Untyped Atomic Integer Binary with Return Data Operation MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Dword Untyped Atomic Integer Binary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 02h Format: Opcode Untyped Atomic Integer Operation message
	13	**Return Data Control**

MSD1R_DWAI2 - Dword Untyped Atomic Integer Binary with Return Data Operation MSD

		Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2R Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Atomic Integer Operation** Format: MDC_AOP2 Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies the Binding Table Index for the message

Dword Untyped Atomic Integer Binary Write Only Operation MSD

MSD1W_DWAI2 - Dword Untyped Atomic Integer Binary Write Only Operation MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Dword Untyped Atomic Integer Binary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 02h Format: Opcode Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 0h

MSD1W_DWAI2 - Dword Untyped Atomic Integer Binary Write Only Operation MSD

		Format: Opcode Specifies that no return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2R Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Atomic Integer Operation** Format: MDC_AOP2 Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies the Binding Table Index for the message

Dword Untyped Atomic Integer Ternary with Return Data Operation MSD

MSD1R_DWAI3 - Dword Untyped Atomic Integer Ternary with Return Data Operation MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Dword Untyped Atomic Integer Ternary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 02h Format: Opcode Untyped Atomic Integer Operation message
	13	**Return Data Control**

MSD1R_DWAI3 - Dword Untyped Atomic Integer Ternary with Return Data Operation MSD

		Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2R Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Atomic Integer Operation** Format: MDC_AOP3 Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies the Binding Table Index for the message

Dword Untyped Atomic Integer Ternary Write Only Operation MSD

MSD1W_DWAI3 - Dword Untyped Atomic Integer Ternary Write Only Operation MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Dword Untyped Atomic Integer Ternary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 02h Format: Opcode Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 0h

MSD1W_DWAI3 - Dword Untyped Atomic Integer Ternary Write Only Operation MSD

		Format: Opcode Specifies that no return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2R Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Atomic Integer Operation** Format: MDC_AOP3 Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies the Binding Table Index for the message

Dword Untyped Atomic Integer Unary with Return Data Operation MSD

MSD1R_DWAI1 - Dword Untyped Atomic Integer Unary with Return Data Operation MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Dword Untyped Atomic Integer Unary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 02h Format: Opcode Untyped Atomic Integer Operation message
	13	**Return Data Control**

MSD1R_DWAI1 - Dword Untyped Atomic Integer Unary with Return Data Operation MSD

		Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2R Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Atomic Integer Operation** Format: MDC_AOP1 Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies the Binding Table Index for the message

Dword Untyped Atomic Integer Unary Write Only Operation MSD

MSD1W_DWAI1 - Dword Untyped Atomic Integer Unary Write Only Operation MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Atomic Operation
Group: Dword Untyped Atomic Integer Unary Operation

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 02h Format: Opcode Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 0h

MSD1W_DWAI1 - Dword Untyped Atomic Integer Unary Write Only Operation MSD

		Format: Opcode Specifies that no return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2R Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Atomic Integer Operation** Format: MDC_AOP1 Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies the Binding Table Index for the message

Else

else - Else

Source: Eulsa

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 prior to entering the if/endif block remain inactive throughout the entire block. All enabled channels upon arriving the else instruction will be 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 matching endif instruction for that conditional block. The following table describes the 32-bit <JIP>. In GEN binary, <JIP> is at location <src1> and must be of type D (signed dword integer). <JIP> must be an immediate operand, it is a signed 32-bit number and is intended to be forward referencing. This value is added to IP pre-increment. If the <branch_ctrl> bit is set, then the <JIP> points to the first join instruction within the else block and <UIP> points to the endif instruction. If the <branch_ctrl> bit is not set, <JIP> and <UIP>, both point to endif.

Format:

```
else (exec_size) JIP  UIP  branch_ctrl
```

Restriction

Predication is not allowed.

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

The branch_ctrl must be set equal to (JIP != UIP).

Syntax

else (exec_size) imm32 imm32 branch_ctrl

Pseudocode

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

Predication	Conditional Modifier	Saturation	Source Modifier	Syntax
N	N	N	N	Unspecified

else - Else

DWord	Bit	Description
0..3	127:96	**JIP** Format: S31 The byte-aligned jump distance if a jump is taken for the channel.
	95:64	**UIP** Format: S31 The byte aligned jump distance if a jump is taken for the instruction.
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

End If

endif - End If

Source: Eulsa

Length Bias: 4

Description

The endif instruction terminates an if/else/endif block of code. It restores execution to the channels that were active prior to the if/else/endif block. The endif instruction is also used to hop out of nested conditionals by jumping to the end of the next outer conditional block when all channels are disabled.

The following table describes the 32-bit JIP. In GEN binary, JIP is at location src1 and must be of type D (signed DWord integer). JIP must be an immediate operand, it is a signed 32-bit number. This value is added to IP pre-increment.

Format:

```
endif JIP
```

Restriction

Predication is not allowed.

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

Syntax

endif (exec_size) imm32

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	127:96	**JIP** Format: S31 Jump Target Offset. The relative offset in bytes if a jump is taken for the instruction.
	95	**Source 0 Address Immediate [9] Sign Bit**

endif - End If

	94:91	**Src1.SrcType** Format: **SrcType**
	90:89	**Src1.RegFile** Format: **RegFile**
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align16') Format: **EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN16**
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1') Format: **EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN1**
	63:32	**Operand Control** Format: **EU_INSTRUCTION_OPERAND_CONTROLS**
	31:0	**Header** Format: **EU_INSTRUCTION_HEADER**

Extended Math Function

math - Extended Math Function

Source: Eulsa

Length Bias: 4

Description
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.<FC> (exec_size) dst src0 src1`

Restriction
Accumulator access is allowed only for IEEE macro functions (invm and rsqtm).
The math instruction does not support indirect addressing modes.
The only supported rounding mode for math instruction is Round to Nearest Even.
INT DIV function does not support SIMD16. INT DIV function does not support simultaneous write to two registers. INT DIV function does not support source modifiers.
The FDIV function is not supported in ALT_MODE.
The math instruction can use GRF or immediates as source. The source formats for immediates must be one of the source formats supported by math operation.
DepCtrl must not be used.
The math instruction must use GRF as destination.
The supported regioning mode for math instruction is align1 and align16. The following restrictions apply for align1 mode: Scalar source is supported. Source and destination horizontal stride must be the same. Regioning must ensure Src.Vstride = Src.Width * Src.Hstride . Source and destination offset must be the same, except the case of scalar source.
Half-float denorms are always retained.
Math Operation rules when float and half-floats are mixed between source or between source and destination operands. The half-float operand must be interleaved in the register for align1 and the source and destination register offset must be be the same to DW granularity. For align16, the half-float operand is allowed to be packed.
The execution size must be no more than 8 when half-floats are used in source or destination operand.
The source operand must not span two registers if the destination operand spans just one register Example: Case (a) // Allowed. The source must be strided by 2. the offset is allowed to select between lower/upper 16b math.inv (8) r10:f r11.0<16;8,2>:hf math.inv (8) r10:f r11.1<16;8,2>:hf math.invm (8) r10:f r11.0<16;8,2>:hf r12.1<16;8,2>:hf Case (b) // Allowed. The destination must be strided by 2. The offset is allowed to select between lower/upper 16b math.inv (8) r10.0<2>:hf r11.0<8;8,1>:f math.inv (8) r10.1<2>:hf r11.0<8;8,1>:f

math - Extended Math Function

math.invm (8) r10.0<2>:hf r11.0<16;8,2>:hf r12.0<16;8,2>:hf Case (c) // Allowed. Destination has stride of 2. The offset is allowed to select between uppoer/lower 16b math.invm (8) r10.0<2>:hf r11.0<8;8,1>:f r12.1<16;8,2>:hf math.invm (8) r10.1<2>:hf r11.1<16;8,2>:hf r12.0<8;8,1>:f Case (d) // Not Allowed. Destination is half-float but is not interleaved. math.inv (8) r10.0<1>:hf r11.0<8;8,1>:f Case (e) // Not Allowed. Source is half-float but not interleaved math.invm (8) r10.0<2>:hf r11.0<8;8,1>:f r12.0<8;8,1>:hf Case (f) // Not Allowed. Source operand spans 2 registers while destination spans one register. math.sin (8) r83.8<1>:hf r12.4<4;4,1>:f

Math Operation rules when half-floats are used on both source and destination operands. The execution size must be 8. The half-float source must be packed or interleaved. When interleaving, both source and destination must be interleaved. Example: Case (a) // Allowed. The source and destination are packed or interleaved math.inv (8) r10.0:hf r11.0<8;8,1>:hf math.inv (8) r10.0<2>:hf r11.0<16;8,2>:hf math.inv (8) r10.8:hf r11.0<8;8,1>:hf math.inv (8) r10.8<2>:hf r11.0<16;8,2>:hf

For one source math operations src1 must encode the null register.

Syntax

```
[(pred)] math.<FC> (exec_size) reg reg reg
```

Pseudocode

```
Evaluate(WrEn);
for (n = 0; n < exec_size; n++) {
    if (WrEn.channel[n] == 1) {
        switch FC[3:0] {
            case 1h: // math.inv
                dst.channel[n] = rcp(src0.channel[n]);
            case 2h: // math.log
                dst.channel[n] = log(src0.channel[n]);
            case 3h: // math.exp
                dst.channel[n] = exp(src0.channel[n]);
            case 4h: // math.sqt
                dst.channel[n] = sqt(src0.channel[n]);
            case 5h: // math.rsqt
                dst.channel[n] = rsqt(src0.channel[n]);
            case 6h: // math.sin
                dst.channel[n] = sin(src0.channel[n]);
            case 7h: // math.cos
                dst.channel[n] = cos(src0.channel[n]);
            case 9h: // math.fdiv  (src0 / src1)
                dst.channel[n] = fdiv(src0.channel[n], src1.channel[n]);
            case Ah: // math.pow
                dst.channel[n] = pow(src0.channel[n], src1/channel[n]);
            case Bh: // math.idiv (src0 / src1)
                idiv(src0.channel[n], src1.channel[n]);
                dst.channel[n] = quotient;
                dst+1.channel[n] = remainder;
            case Ch: // math.iqot
                idiv(src0.channel[n], src1.channel[n]);
                dst.channel[n] = quotient;
            case Dh: // math.irem
                idiv(src0.channel[n], src1.channel[n]);
                dst.channel[n] = remainder;
            case Eh: // math.invm
                dst.channel[n] = invm(src0.channel[n], src1.channel[n]);
```

math - Extended Math Function

```
                case Fh: // math.rsqtm
                    dst.channel[n] = rsqtm(src0.channel[n]);
                }
            }
        }
    }
```

Predication	Conditional Modifier	Saturation	Source Modifier []	Syntax	Subfunctions []
Y	N	Y	Y	Unspecified	Unspecified

Src Types	Dst Types
F	F
D	D
UD	UD
F, HF	F, HF

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

Find First Bit from LSB Side

fbl - Find First Bit from LSB Side

Source: Eulsa

Length Bias: 4

The fbl instruction counts component-wise the number of LSB 0 bits before the first 1 bit in src0, storing that number in dst.

Format:

```
[(pred)] fbl (exec_size) dst src0
```

Programming Notes

If src0 contains no 1 bits, store 0xFFFFFFFF in dst.

Restriction

No accumulator access, implicit or explicit.

Syntax

```
[(pred)] fbl (exec_size) reg reg
[(pred)] fbl (exec_size) reg imm32
```

Pseudocode

```
Evaluate(WrEn);
for ( n = 0; n < exec_size; n++ ) {
    if ( WrEn.chan[n] ) {
        UD cnt = 0;
        UD udScalar = src0.chan[n];
        while ( (udScalar & 1) == 0 && cnt != 32 ) {
            cnt ++;
            udScalar = udScalar » 1;
        }
        if ( src0.chan[n] == 0x00000000 ) {
            dst.chan[n] = 0xFFFFFFFF;
        }
        else {
            dst.chan[n] = cnt;
        }
    }
}
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

Src Types	Dst Types
UD	UD

DWord	Bit	Description

fbl - Find First Bit from LSB Side		
0..3	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: **EU_INSTRUCTION_SOURCES_REG**
	127:64	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: **EU_INSTRUCTION_SOURCES_IMM32**
	63:32	**Operand Controls** Format: **EU_INSTRUCTION_OPERAND_CONTROLS**
	31:0	**Header** Format: **EU_INSTRUCTION_HEADER**

Find First Bit from MSB Side

fbh - Find First Bit from MSB Side

Source: Eulsa

Length Bias: 4

If src0 is unsigned, the fbh instruction counts component-wise the leading zeros from src0 and stores the resulting counts in dst. If src0 is signed and positive, the fbh instruction counts component-wise the leading zeros from src0 and stores the resulting counts in dst. If src0 is signed and negative, the fbh instruction counts component-wise the leading ones from src0 and stores the resulting counts in dst.

Format:

```
[(pred)] fbh (exec_size) dst src0
```

Programming Notes

If src0 is zero, store 0xFFFFFFFF in dst.

If src0 is signed and is -1 (0xFFFFFFFF), store 0xFFFFFFFF in dst.

Restriction

No accumulator access, implicit or explicit.

Syntax

```
[(pred)] fbh (exec_size) reg reg
[(pred)] fbh (exec_size) reg imm32
```

Pseudocode

```
Evaluate(WrEn);
      for ( n = 0; n < exec_size; n++ ) {
         if ( WrEn.chan[n] ) {
            UD cnt = 0;
            if ( src0 is unsigned ) {
               UD udScalar = src0.chan[n];
               while ( (udScalar & (1 « 31)) == 0 && cnt != 32 ) {
                  cnt ++;
                  udScalar = udScalar « 1;
               }
               if ( src0.chan[n] == 0x00000000 ) {
                  dst.chan[n] = 0xFFFFFFFF;
               }
               else {
                  dst.chan[n] = cnt;
               }
            }
            else {  // src0 is signed.
               D dScalar = src0.chan[n];
               bit cval = dScalar[31];
               while ((dScalar & (1 « 31)) == cval && cnt != 32 ) {
                  cnt ++;
```

fbh - Find First Bit from MSB Side

```
                    dScalar = dScalar « 1;
                }
                if ( (src0.chan[n] == 0xFFFFFFFF) || (src0.chan[n] == 0x00000000) ) {
                    dst.chan[n] = 0xFFFFFFFF;
                }
                else {
                    dst.chan[n] = cnt;
                }
            }
        }
    }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

Src Types	Dst Types
D, UD	UD

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

Fraction

frc - Fraction

Source: Eulsa

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

Goto

goto - Goto

Source: Eulsa

Length Bias: 4

The goto instruction directs the instruction pointer to the offset specified by the UIP offset or to the next IP based on the BranchCtrl bit in the instruction. The active channels that are predicated on this instruction will take the IP + UIP path when BranchCtrl is set else the channels take IP + 1. The active channels that are not predicated on this instruction will be made inactive and waiting to be joined at the join IP. The join IP is IP + UIP when BranchCtrl is clear else it is the next IP.
When there are no active channels the instruction pointer will move to IP + JIP.
The goto instruction is used in conjunction with a join instruction. A goto deactivates some channels that are reactivated at some program-specified join instruction. See the join instruction for the activation rules.
The goto and join instructions enable unstructured program control flow. These instructions must be used with additional care where dangling channels can result without proper compiler checks, meaning that it is expected that programs will navigate through these paths to reactivate the channels. Hardware does not provide native checks or reconvergence.
The following table describes the two 32-bit instruction pointer offsets. Both the JIP and UIP are signed 32-bit numbers, added to IP pre-increment. In GEN binary, JIP and UIP are at locations src0 and src1 and must be of type DW (signed DWord integer).
If SPF is ON, none of the PcIP are updated.

Format:

```
[(pred)] goto (exec_size) JIP UIP branch_ctrl
```

Restriction

Cannot have a goto in the body and the corresponding join in the function or the subroutine. Similarly the brd and brc.

Syntax

```
[(pred)] goto (exec_size) imm32 imm32 branch_ctrl
```

Pseudocode

Evaluate(WrEn);

```
 for ( n = 0; n < exec_size; n++ ) { if ( WrEn.chan[n] ) { // for the predicated
active channels PcIP[n] = IP + UIP; } else { // join IP, for the active non
predicated channels PcIP[n] = IP + 1; } } if ( BranchCtrl ) { // if (PcIP != (IP
+ UIP) ) { // for all channels if (PcIP != (IP + 1) ) { // for all channels
Jump(IP + JIP); } else { Jump(IP + 1); } } else { Jump(IP + UIP); } } else { //
if (PcIP != (IP + 1) ) { // for all channels Jump(IP + JIP); } else { Jump(IP +
1); } }
```

Predication	Conditional Modifier	Saturation	Source Modifier

goto - Goto

Y	N	N	N

DWord	Bit	Description
0..3	127:96	**JIP** Format: S31 The byte-aligned jump distance if a jump is taken for the channel.
	95:64	**UIP** Format: S31 The byte aligned jump distance if a jump is taken for the instruction.
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

GPGPU_CSR_BASE_ADDRESS

GPGPU_CSR_BASE_ADDRESS

Source: BSpec

Length Bias: 2

The GPGPU_CSR_BASE_ADDRESS command sets the base pointers for EU and L3 to Context Save and Restore EU State and SLM for **GPGPU** mid-thread preemption.

Programming Notes

Execution of this command causes a full pipeline flush, thus its use should be minimized for higher performance. State and instruction caches are flushed on completion of the flush.

SW must always program PIPE_CONTROL with "CS Stall" and "Render Target Cache Flush Enable" set prior to programming GPGPU_CSR_BASE_ADDRESS command for GPGPU workloads i.e when pipeline select is GPGPU via PIPELINE_SELECT command. This is required to achieve better GPGPU preemption latencies for certain programming sequences. If programming PIPE_CONTROL has performance implications then preemption latencies can be trade off against performance by not implementing this programming note.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: Opcode
	28:27	**Command SubType** Default Value: 0h GFXPIPE_COMMON Format: Opcode
	26:24	**3D Command Opcode** Default Value: 1h GFXPIPE_NONPIPELINED Format: Opcode
	23:16	**3D Command Sub Opcode** Default Value: 04h GPGPU_CSR_BASE_ADDRESS Format: Opcode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Total Length -2 Value / Name / Description: 1h / Length_1 **[Default]** / Excludes DWord(0,1)
1..2	63:12	**GPGPU CSR Base Address**

GPGPU_CSR_BASE_ADDRESS

		Format: GraphicsAddress[63:12] Specifies the 256K-byte aligned base address for GPGPU context GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].
	11:0	**Reserved** Format: MBZ

GPGPU_WALKER

GPGPU_WALKER

Source: RenderCS

Length Bias: 2

Programming Notes

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h GPGPU_WALKER Format: OpCode
	23:16	**SubOpcode** Default Value: 05h GPGPU_WALKER SubOp Format: OpCode
	15:11	**Reserved** Format: MBZ
	10	**Indirect Parameter Enable** Format: Enable If set, the values in DW 7, 10, 12 are ignored and replaced by the current values of the corresponding GPGPU_xxx MMIO registers: • GPGPU_DISPATCHDIMX (instead of DW7) • GPGPU_DISPATCHDIMY (instead of DW10) • GPGPU_DISPATCHDIMZ (instead of DW12)
	9	**Reserved** Format: MBZ
	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** Format: =n Total Length - 2 **Value** / **Name** / **Description** 0Dh / DWORD_COUNT_n **[Default]** / Excludes DWord (0,1)
1	31:8	**Reserved**
	7:6	**Reserved** Format: MBZ
	5:0	**Interface Descriptor Offset** Format: U6 This field specifies the offset from the interface descriptor base pointer to the interface descriptor which will be applied to this object. It is specified in units of interface descriptors.
2	31:17	**Reserved** Format: MBZ
	16:0	**Indirect Data Length** Format: U17 in bytes This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. If Indirect Data is enabled, it is assumed that CURBE is not being used. This field must have the same alignment as the Indirect Object Data Start Address. It must be DQWord (32-byte) aligned.
3	31:6	**Indirect Data Start Address** Format: IndirectObjectOffset[31:6] This field specifies the Graphics Memory starting address of the data to be loaded into the kernel for processing. This pointer is relative to the **Indirect Object Base Address.** Hardware ignores this field if indirect data is not present. Alignment of this address depends on the mode of operation. It is the 64-byte aligned address of the indirect data. **Value** / **Name** / **Description** [0x0,0x3FFFFFF] / / Range 0 .. 512MB. (Bits 31:29 MBZ)
	5:0	**Reserved** Format: MBZ
4	31:30	**SIMD Size** This field determines the size of the payload and the number of bits of the execution mask that

		are expected. The kernel pointed to by the interface descriptor should match the SIMD declared here. **Value** / **Name** / **Description** 0 / SIMD8 / 8 LSBs of the execution mask are used 1 / SIMD16 / 16 LSBs used in execution mask 2 / SIMD32 / 32 bits of execution mask used
	29:22	**Reserved** Format: MBZ
	21:16	**Thread Depth Counter Maximum** The maximum value of the thread depth counter. Since the counter starts at 0, the depth is this value + 1. **(Thread_Depth_Max+1)*(Thread_Height_Max+1)*(Thread_Width_Max+1)** must equal **Number of Threads in GPGPU Thread Group** in the Interface Descriptor.
	15:14	**Reserved** Format: MBZ
	13:8	**Thread Height Counter Maximum** Format: U6-1 The maximum value of the thread height counter. The height is this value + 1.
	7:6	**Reserved** Format: MBZ
	5:0	**Thread Width Counter Maximum** Format: U6-1 The maximum value of the thread width counter. The width is this value + 1.
5	31:0	**Thread Group ID Starting X** This is the initial value of the X component of the thread group. When X reaches the maximum value it rolls around to this value.
6	31:0	**Reserved** Format: MBZ
7	31:0	**Thread Group ID X Dimension** The X dimension of the thread group (maximum X is dimension -1)
8	31:0	**Thread Group ID Starting Y** This is the initial value of the Y component of the thread group. When Y reaches the maximum value it rolls around to this value.
9	31:0	**Reserved** Format: MBZ
10	31:0	**Thread Group ID Y Dimension** The Y dimension of the thread group (maximum Y is dimension -1)

GPGPU_WALKER		
11	31:0	**Thread Group ID Starting/Resume Z** This is the initial value of the Z component of the thread group.
12	31:0	**Thread Group ID Z Dimension** The Z dimension of the thread group (maximum Z is dimension -1)
13	31:0	**Right Execution Mask** The execution mask used for threads on the right (maximum X) of the thread group.
14	31:0	**Bottom Execution Mask** The execution mask used for threads on the bottom (maximum Y) of the thread group.

Half Precision HI8DS Render Target Write MSD

MSD_RTWH_HI8DS - Half Precision HI8DS Render Target Write MSD

Source: EuSubFunctionRenderDataPort
Length Bias: 1
Family: Other
Group: Render Target R/W

DWord	Bit	Description
0	31	**Reserved** Format: MBZ Ignored
	30	**Message Precision Subtype** Default Value: 1h Format: Opcode Half precision data message
	29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.

MSD_RTWH_HI8DS - Half Precision HI8DS Render Target Write MSD

	19	**Header Present** | | | | Format: | MDC_MHP | If set, indicates that the message includes the 2-register header.
	18	**Per-Coarse Pixel PS outputs enable** | | | | Format: | Enable | This bit indicates the render target write is a coarse pixel write. **Programming Notes** This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.
	17:14	**Message Type** | Default Value: | 0Ch | | | | | Format: | Opcode | Render Target Write message
	13	**Per-Sample PS Enable** | | | | Format: | Enable | If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot. **Programming Notes** This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE. When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0. When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples). When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.

MSD_RTWH_HI8DS - Half Precision HI8DS Render Target Write MSD

	12	**Last Render Target Select** Format: Enable This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.
	11	**Slot Group Select** Format: MDC_RT_SGS This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
	10:8	**Render Target Message Subtype** Default Value: 3h Format: Opcode SIMD8 dual source message. Use slots [15:8] for pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [31:24] are referenced instead of [15:8].
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Half Precision LO8DS Render Target Write MSD

MSD_RTWH_LO8DS - Half Precision LO8DS Render Target Write MSD

Source: EuSubFunctionRenderDataPort
Length Bias: 1
Family: Other
Group: Render Target R/W

DWord	Bit	Description
0	31	**Reserved** Format: MBZ Ignored
	30	**Message Precision Subtype** Default Value: 1h Format: Opcode Half precision data message
	29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.

MSD_RTWH_LO8DS - Half Precision LO8DS Render Target Write MSD

	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the 2-register header.
	18	**Per-Coarse Pixel PS outputs enable** Format: Enable This bit indicates the render target write is a coarse pixel write. **Programming Notes** This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.
	17:14	**Message Type** Default Value: 0Ch Format: Opcode Render Target Write message
	13	**Per-Sample PS Enable** Format: Enable If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot. **Programming Notes** This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE. When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0. When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples). When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.

MSD_RTWH_LO8DS - Half Precision LO8DS Render Target Write MSD

	12	**Last Render Target Select** Format: Enable This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.
	11	**Slot Group Select** Format: MDC_RT_SGS This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
	10:8	**Render Target Message Subtype** Default Value: 2h Format: Opcode SIMD8 dual source message. Use slots [7:0] for pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [23:16] are referenced instead of [7:0].
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Half Precision REP16 Render Target Write MSD

MSD_RTWH_REP16 - Half Precision REP16 Render Target Write MSD

Source: EuSubFunctionRenderDataPort
Length Bias: 1
Family: Other
Group: Render Target R/W

DWord	Bit	Description
0	31	**Reserved** Format: MBZ Ignored
	30	**Message Precision Subtype** Default Value: 1h Format: Opcode Half precision data message
	29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.

MSD_RTWH_REP16 - Half Precision REP16 Render Target Write MSD

	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the 2-register header.
	18	**Per-Coarse Pixel PS outputs enable** Format: Enable This bit indicates the render target write is a coarse pixel write. **Programming Notes** This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.
	17:14	**Message Type** Default Value: 0Ch Format: Opcode Render Target Write message
	13	**Per-Sample PS Enable** Format: Enable If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot. **Programming Notes** This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE. When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0. When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples). When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.

MSD_RTWH_REP16 - Half Precision REP16 Render Target Write MSD

	12	**Last Render Target Select** Format: Enable This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.
	11	**Slot Group Select** Format: MDC_RT_SGS This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
	10:8	**Render Target Message Subtype** Default Value: 1h Format: Opcode SIMD16 Single source message with replicated data. Use slots [15:0] for pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [31:16] are referenced instead of [15:0].
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Half Precision SIMD8 Render Target Write MSD

MSD_RTWH_SIMD8 - Half Precision SIMD8 Render Target Write MSD

Source: EuSubFunctionRenderDataPort
Length Bias: 1
Family: Other
Group: Render Target R/W

DWord	Bit	Description
0	31	**Reserved** Format: MBZ Ignored
	30	**Message Precision Subtype** Default Value: 1h Format: Opcode Half precision data message
	29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.

MSD_RTWH_SIMD8 - Half Precision SIMD8 Render Target Write MSD

	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the 2-register header.
	18	**Per-Coarse Pixel PS outputs enable** Format: Enable This bit indicates the render target write is a coarse pixel write. **Programming Notes** This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.
	17:14	**Message Type** Default Value: 0Ch Format: Opcode Render Target Write message
	13	**Per-Sample PS Enable** Format: Enable If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot. **Programming Notes** This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE. When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0. When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples). When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.

MSD_RTWH_SIMD8 - Half Precision SIMD8 Render Target Write MSD

	12	**Last Render Target Select** Format: Enable This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.
	11	**Slot Group Select** Format: MDC_RT_SGS This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
	10:8	**Render Target Message Subtype** Default Value: 4h Format: Opcode SIMD8 single source message. Use slots [7:0] for pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [23:16] are referenced instead of [7:0].
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Half Precision SIMD16 Render Target Write MSD

MSD_RTWH_SIMD16 - Half Precision SIMD16 Render Target Write MSD

Source: EuSubFunctionRenderDataPort
Length Bias: 1
Family: Other
Group: Render Target R/W

DWord	Bit	Description
0	31	**Reserved** Format: MBZ Ignored
	30	**Message Precision Subtype** Default Value: 1h Format: Opcode Half precision data message
	29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.

MSD_RTWH_SIMD16 - Half Precision SIMD16 Render Target Write MSD

	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the 2-register header.
	18	**Per-Coarse Pixel PS outputs enable** Format: Enable This bit indicates the render target write is a coarse pixel write. **Programming Notes** This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.
	17:14	**Message Type** Default Value: 0Ch Format: Opcode Render Target Write message
	13	**Per-Sample PS Enable** Format: Enable If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot. **Programming Notes** This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE. When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0. When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples). When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.

MSD_RTWH_SIMD16 - Half Precision SIMD16 Render Target Write MSD

	12	**Last Render Target Select** Format: Enable This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.
	11	**Slot Group Select** Format: MDC_RT_SGS This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
	10:8	**Render Target Message Subtype** Default Value: 0h Format: Opcode SIMD16 Single source message. Use slots [15:0] for pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [31:16] are referenced instead of [15:0].
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Halt

halt - Halt

Source: Eulsa

Length Bias: 4

Description

The halt instruction temporarily suspends execution for all enabled compute channels. Upon execution, the enabled channels are sent to the instruction at (IP + UIP), if all channels are enabled at HALT, jump to the instruction at (IP + JIP). If the halt instruction is not inside any conditional code block, the values of JIP and UIP should be the same. If the halt instruction is inside a conditional code block, the UIP should be the end of the program and the JIP should be the end of the inner most conditional code block. The UIP must point to a HALT Instruction. If SPF is ON, the UIP must be used to update IP; JIP is not used in this case.

The following table describes the two 32-bit instruction pointer offsets. Both the JIP and UIP are signed 32-bit numbers, added to IP pre-increment. In GEN binary, JIP and UIP are at locations src0 and src1 and must be of type DW (signed DWord integer). When the offsets are immediate, src0 regfile must be immediate and dst must be null.

Format:

```
[(pred)] halt (exec_size) JIP UIP
```

Syntax

```
[(pred)] halt (exec_size) imm32 imm32
```

Pseudocode

```
Evaluate(WrEn);
          for ( n = 0; n < 32; n++ ) {
             if ( WrEn.channel[n] ) {
                PcIP[n] = IP + UIP;
             else {
                PcIP[n] = IP + 1;
             }
          }

          if ( PcIP != (IP + 1) ) {  // for all channels
             Jump(IP + JIP);
          }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

DWord	Bit	Description
0..3	127:96	**JIP**

halt - Halt

		Format: S31 The byte-aligned jump distance if a jump is taken for the channel.
	95:64	**UIP** Format: S31 The byte aligned jump distance if a jump is taken for the instruction.
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

HCP_BSD_OBJECT

HCP_BSD_OBJECT

Source: VideoCS

Length Bias: 2

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

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

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

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

HCP_BSD_OBJECT

1	31:0	**Indirect BSD Data Length** Format: U32 Specifies the length in bytes of the bitstream data for the current slice. It includes the first byte of the slice and the last non-zero byte of the in the slice. Specifically, the zero-padding bytes (if present) and the next start-code are excluded. **Value** / **Name** / **Description** [0,268435455] / Data_Length_with_28_bits_only / Valid range is only from 0 to 268435455, which is corresponding to lower 28 bits. This restriction is for old project which only use 28 bits data length.
2	31:29	**Reserved** Format: MBZ
	28:0	**Indirect Data Start Address** Format: U29 Specifies the byte-aligned graphics memory starting address of the slice bit stream relative to the **BSD Indirect Object Base Address**.

HCP_FQM_STATE

HCP_FQM_STATE

Source: VideoCS

Length Bias: 2

The HCP_FQM_STATE command loads the custom HEVC quantization tables into local RAM and may be issued up to 8 times: 4 scaling list per intra and inter.

Driver is responsible for performing the Scaling List division. So, save the division HW cost in HW. The 1/x value is provided in 16-bit fixed-point precision as ((1«17)/QM +1) » 1. .

Note: FQM is computed as (2^16)/QM. If QM=1, FQM=all 1's.

To simplify the design, only a limited number of scaling lists are provided at the PAK interface: default two SizeID0 and two SizeID123 (one set for inter and the other set for intra), and the encoder only allows custom entries for these four matrices. The DC value of SizeID2 and SizeID3 will be provided.

When the scaling_list_enable_flag is set to disable, the scaling matrix is still sent to the PAK, and with all entries programmed to the same value of 16.

This is a picture level state command and is issued in encoding processes only.

Dwords 2-33 form a table for the DCT coefficients, 2 16-bit coefficients/DWord.

- Size 4x4 for SizeID0, DWords 2-9.
- Size 8x8 for SizeID1/2/3, DWords 2-33.

SizeID 0 (Table 4-13)

4x4	[31:16]	[15:0]
DWord 2	AC(0,1)	DC
DWord 3	AC(0,3)	AC(0,2)
DWord 4	AC(1,1)	AC(1,0)
DWord 5	AC(1,3)	AC(1,2)
DWord 6	AC(2,1)	AC(2,0)
DWord 7	AC(2,3)	AC(2,2)
DWord 8	AC(3,1)	AC(3,0)
DWord 9	AC(3,3)	AC(3,2)

SizeID 1, 2, 3 (Table 4-14)

8x8	[31:16]	[15:0]
DWord 2	AC(0,1)	DC
DWord 3	AC(0,3)	AC(0,2)
DWord 4	AC(0,5)	AC(0,4)
DWord 5	AC(0,7)	AC(0,6)

HCP_FQM_STATE

DWord 6	AC(1,1)	AC(1,0)
DWord 7	AC(1,3)	AC(1,2)
DWord 8	AC(1,5)	AC(1,4)
DWord 9	AC(1,7)	AC(1,6)
...		
DWord 30	AC(7,1)	AC(7,0)
DWord 31	AC(7,3)	AC(7,2)
DWord 32	AC(7,5)	AC(7,4)
DWord 33	AC(7,7)	AC(7,6)

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: 5h HCP_FQM_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 20h; Name:
1	31:16	**FQM DC Value: (1/DC):** Format: U16 Specifies DC value of the scaling list for 16x16 (SizeID=2) or 32x32 (SizeID=3).

		DC Value = scaling_list_dc_coef_minus8 + 8. Driver will do the division.
	15:5	**Reserved** Format: U11
	4:3	**Color Component** Format: U2 Luma and Chroma's share the same scaling list and DC value for the same SizeID. Value / Name: 0 Luma 1 Chroma Cb 2 Chroma Cr 3 Reserved
	2:1	**SizeID** Format: U2 Value / Name: 0 SizeID 0 4x4 1 SizeID 1, 2, 3 (8x8, 16x16, 32x32) 2 SizeID 2 (for DC value in 16x16) 3 SizeID 3 (for DC value in 32x32)
	0	**Intra/Inter** Format: U1 This field specifies the quant matrix intra or inter type. Value / Name: 0 Intra 1 Inter
2..33	1023:0	**QuantizerMatrix**

HCP_IND_OBJ_BASE_ADDR_STATE

HCP_IND_OBJ_BASE_ADDR_STATE

Source: VideoCS

Length Bias: 2

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

The HCP_IND_OBJ_BASE_ADDR_STATE command is used to define the indirect object base address of the stream in graphics memory. This is a frame level command.
This is a picture level state command and is issued in both encoding and decoding processes.

Compressed Header Format

Fields	Bits	
Bin	0	Kernel Binarized Syntax
Probability select	1	0 ->indicates probability 128 1 ->indicates probability 256
	Repeat to pack a Cacheline	

Partition1 and TileSize record

Fields	Bits	
Tile Size	31:0	Partition1 Size is 16-bit value, Tile Size is 32-bit value
AddressOffset	63:32	Cacheline Address Offset to be Modified
Offset	69:64	Byte offset to be Modified
16-bit vs 32-bit update	70	0: Update 16-bit; 1: Update 32-bit
Reserved	511:71	

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h

HCP_IND_OBJ_BASE_ADDR_STATE

	22:16	**Media Instruction Command** Default Value: 3h HCP_IND_OBJ_BASE_ADDR_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: Ch — Name:
1..2	63:0	**HCP Indirect Bitstream Object Base Address** Format: **SplitBaseAddress4KByteAligned** Specifies the 4K-byte aligned memory base address for the read-only indirect data object pointed in the HCP_BSD_OBJECT command for fetching (reading) the compressed Slice Data.
3	31:0	**HCP Indirect Bitstream Object Memory Address Attributes** Format: **MemoryAddressAttributes**
4..5	63:0	**HCP Indirect Bitstream Object Access Upper Bound** Format: **SplitBaseAddress4KByteAligned** Decoder only. This field specifies the 4K-byte aligned maximum memory address access by the indirect data object in the HCP_BSD_OBJECT command for the slice bit stream. Indirect data accessed at this address or greater will cause the HCP to stop issuing requests to the GAC and the BSP VLD will then only receive zeros until a slice done is received. Setting this field to 0 will cause this range to be ignored by the HCP.
6..7	63:0	**HCP Indirect CU Object Base Address** Format: **BaseAddress4KByteAligned** Encoder only. Specifies the 4K-byte aligned data buffer base address for the read-only indirect data object for fetching (reading) per CU data during the encoding process.
8	31:0	**HCP Indirect CU Object Object Memory Address Attributes**

HCP_IND_OBJ_BASE_ADDR_STATE

		Format: **MemoryAddressAttributes**
9..10	63:0	**HCP PAK-BSE Object Base Address** Format: **BaseAddress4KByteAligned** Encoder only. Specifies the 4K-byte aligned memory base address for the write-only data pointed by the PAK engine for writing out the compressed bitstream.
11	31:0	**HCP PAK-BSE Object Address Memory Address Attributes** Format: **MemoryAddressAttributes** Encoder only.
12..13	63:0	**HCP PAK-BSE Object Access Upper Bound** Format: **SplitBaseAddress4KByteAligned** Encoder only. This field specifies the 4K-byte aligned maximum memory address access by the HCP_PAK_OBJECT command for writing out the slice bit stream. Indirect data accessed at this address and beyond will be blocked by the hardware and ignored. This address must be greater than the HCP PAK-BSE Object Base Address state.
14..15	63:0	**HCP VP9 PAK Compressed Header Syntax Streamin- Base Address** Exists If: //Encoder Only Format: **BaseAddress4KByteAligned** Specifies the 4K-byte aligned data buffer base address for the read-only Probability counters fetching during the encoding process..
16	31:0	**HCP VP9 PAK Compressed Header Syntax StreamIn Memory Address Attributes** Format: **MemoryAddressAttributes**
17..18	63:0	**HCP VP9 PAK Probability Counter StreamOut- Base Address** Exists If: //Encoder Only Format: **BaseAddress4KByteAligned** Specifies the 4K-byte aligned data buffer base address for the write-only Probability counters fetching during the encoding process..

HCP_IND_OBJ_BASE_ADDR_STATE

19	31:0	**HCP VP9 PAK Probability Counter StreamOut Memory Address Attributes** Exists If: //Encoder Only Format: **MemoryAddressAttributes**
20..21	63:0	**HCP VP9 PAK Probability Deltas StreamIn- Base Address** Exists If: //Encoder Only Format: **BaseAddress4KByteAligned** Specifies the 4K-byte aligned data buffer base address for the read-only Probability differences during the encoding process.
22	31:0	**HCP VP9 PAK Probability Deltas StreamIn Memory Address Attributes** Exists If: //Encoder Only Format: **MemoryAddressAttributes**
23..24	63:0	**HCP VP9 PAK Tile Record StreamOut- Base Address** Format: **BaseAddress4KByteAligned** Specifies the 4K-byte aligned memory base address for the write-only data pointed by the PAK engine for writing out the Tile Record.
25	31:0	**HCP VP9 PAK Tile Record StreamOut Memory Address Attributes** Exists If: //Encoder Only Format: **MemoryAddressAttributes**
26..27	63:0	**HCP VP9 PAK CU Level Statistic StreamOut- Base Address** Exists If: //Encoder Only Format: **BaseAddress4KByteAligned** Specifies the 4K-byte aligned memory base address for the write-only data pointed by the PAK engine for writing out the CU Record.
28	31:0	**HCP VP9 PAK CU Level Statistic StreamOut Memory Address Attributes** Exists If: //Encoder Only Format: **MemoryAddressAttributes**

HCP_PAK_INSERT_OBJECT

HCP_PAK_INSERT_OBJECT	
Source:	VideoCS
Length Bias:	2

It is an encoder only command, operating at bitstream level, before and after SliceData compressed bitstream. It is setup by the header and tail present flags in the Slice State command. If these flags are set and no subsequent PAK_INSERT_OBJECT commands are issued, the pipeline will hang.
The HCP_ PAK_ INSERT _OBJECT command supports both inline and indirect data payload, but only one can be active at any time. It is issued to insert a chunk of bits (payload) into the current compressed bitstream output buffer (specified in the HCP_PAK-BSE Object Base Address field of the HCP_IND_OBJ_BASE_ADDR_STATE command) starting at its current write pointer bit position. Hardware will keep track of this write pointer's byte position and the associated next bit insertion position index.
It is a variable length command when the payload (data to be inserted) is presented as inline data within the command itself. The inline payload is a multiple of 32-bit (1 DW), as the data bus to the compressed bitstream output buffer is 32-bit wide.
The payload 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. Note that : Stitch Command is used if the beginning position of data is in bit position. When PAK Insert Command is used the beginning position must be in byte position.
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 bitstream.
Internally, HCP 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.
The payload 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.
Data to be inserted can be a valid NAL units or a partial NAL unit. It can be any encoded syntax elements bit data before the encoded Slice Data (PAK Object Command) of the current Slice - SPS NAL, PPS NAL, SEI NAL and Other Non-Slice NAL, Leading_Zero_8_bits (as many bytes as there is), Start Code , Slice Header. Any 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 bitstream, whichever comes first Cabac_Zero_Word or Trailing_Zero_8bits (as many bytes as there is).
Certain NAL unit has a minimum byte size requirement. As such the hardware will optionally (enabled by SLICE 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.

HCP_PAK_INSERT_OBJECT

Context switch interrupt is not supported by this command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: 22h HCP_PAK_INSERT_OBJECT Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Default Value: 0h DWORD_COUNT_n Format: =n (Excludes Dwords 0, 1) =Total Length - 2, soDWord Length = X, where X is in the size of the payload in DWs 2..n which has the range of [1,4095]
1	31	**Indirect Payload Enable** Format: Enable Only one of these two payload modes can be active at any time. When Slice Size Conformance is enable the Payload(header) must be inline only so this bit set to MBZ.

Value	Name	Description
0	inline payload is used	
1	indirect payload is used	

Programming Notes
In VP9 scalability mode, tail should be inserted in inline mode only so this bit Must Be zero.

	30:18	**Reserved** Format: MBZ

17:16 **DataByteOffset - SrcDataStartingByteOffset[1:0]**

Format:	U2

Source Data Starting Byte Position within the very first inline DW.

15 **HeaderLengthExcludeFrmSize**

Format:	U1

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 HCP_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 HCP_PAK_INSERT_OBJECT).

Value	Description
0	All bits accumulated
1	Bits during current call are not accumulated

Value	Name	Description
0	All bits accumulated	
1	Bits during current call are not accumulated	

Programming Notes
Must be set to 0 for JPEG encoder.

14 **Reserved**

Format:	MBZ

13:8 **DataBitsInLastDW - SrCDataEndingBitInclusion[5:0]**

Format:	U6

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.
For VP9:
The Driver has to give byte aligned Data for the last inline DW. (the driver pads Zeros to next byte boundary to the original header if it was not byte aligned on the last inline DW).

Value	Name
[1,32]	

7:4 **SkipEmulByteCnt - Skip Emulation Byte Count**

Format:	U4

Skip emulation check for number of starting bytes It can be programmed from 0 to 15 bytes. For example, to skip the start code that has already prefixed in the bitstream.
Only valid for HEVC and reserved for VP9.

HCP_PAK_INSERT_OBJECT

	3	**EmulationFlag - EmulationByteBitsInsertEnable** Format: U1 Only valid for HEVC and reserved for VP9. **Value** / **Name** / **Description** 0 — When this bit is set to 0, HW will disable the insertion of emulation bytes into the bitstream. 1 — When this bit is set to 1, HW will enable the insertion of emulation bytes into the bitstream.
	2	**LastHeaderFlag - LastSrcHeaderDataInsertCommandFlag** Format: U1 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.
	1	**EndOfSliceFlag - LastDstDataInsertCommandFlag** Format: U1 No more insertion command and no more PAK-OBJECT command follows. Flush data out to memory.
	0	**Reserved** Format: MBZ
2..n	127:0	**Indirect Payload** Exists If: ([Indirect Payload Enable]==1) Format: **HCP_PAK_INSERT_OBJECT_INDIRECT_PAYLOAD**
	31:0	**Inline PayLoad** Exists If: ([Indirect Payload Enable]==0) Format: U32 Actual Data (inline) to be inserted to the output bitstream buffer.

HCP_PAK_OBJECT

HCP_PAK_OBJECT

Source: VideoCS

Length Bias: 2

This is a per-LCU command for encoder only.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: 21h HCP_PAK_OBJECT Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 1h — Name:
1	31	**LastCtbOfSlice Flag** Format: Boolean 0 — False 1 — True
	30	**LastCtbOfTile Flag** Format: U1

HCP_PAK_OBJECT		
		Indicates last LCU or SB of a Tile
	29:24	**CU_count_minus1** Format: U6 Number of CUs in the current LCU = CU_count_minus1 + 1. Minimum, there must be 1 CU in a LCU.
	23:21	**Reserved** Format: MBZ
	20:0	**split_coding_unit_flag[x0][y0]** Format: **Split_coding_unit_flags** If CU size=64x64 and DQP!=0, split should happen at least once
2	31:16	**Current LCU Y Addr** Format: U16
	15:0	**Current LCU X Addr** Format: U16

HCP_PIC_STATE

HCP_PIC_STATE

Source: VideoCS

Length Bias: 2

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

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: 10h HCP_PIC_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 1Dh — Name:
1	31:27	**Reserved** Format: MBZ
	26:16	**FrameHeightInMinCbMinus1**

HCP_PIC_STATE

		Format: U11 Specifies the height of each decoded picture in units of minimum coding block size. **Programming Notes** The decoded picture height in units of luma samples equals • (FrameHeightInMinCbMinus1 + 1) * • (1 « (log2_min_coding_block_size_minus3 + 3)) • The maximum support picture size is 16384 pixels for both encoder and decoder. The decoded picture height in units of luma samples equals (FrameHeightInMinCbMinus1 + 1) * (1 « (log2_min_coding_block_size_minus3 + 3)) The maximum support picture size is 16384 pixels. In HEVC Encoder mode, the following restrictions apply. Note : for a frame width that is not an integer multiple of LCU : Kernel : on last LCU at frame's right edge, it must align to CU boundary. This applies to all size of LCU: 16x16, 32x32, 64x64. Kernel does not count/include CUs outside of the picture in PakObj/CU_record respectively. Driver : sets up a LCU aligned (both in X/Y direction) surface.
	15	**PAK Transform Skip Enable** If global bit, **transform_skip_enabled_flag** in image state dw4.22 set to 1 and this bit set to, 1 -> HW will perform transform skip and over write the ENC decision in CU record 0-> HW will not perform transform skip if **transform_skip_enabled_flag=0**, no transform skip performed in HW or ENC **Programming Notes** Encoder Only
	14:11	**Reserved** Format: MBZ
	10:0	**FrameWidthInMinCbMinus1** Format: U11 Specifies the width of each decoded picture in units of minimum coding block size. **Programming Notes** The decoded picture width in units of luma samples equals (FrameWidthInMinCbMinus1 + 1) * (1 « (log2_min_coding_block_size_minus3 + 3)) The maximum support picture size is 16384pixels.

HCP_PIC_STATE

2 — 31:29 — Chroma Subsampling

Description
Specify the chroma subsampling of the current bitstream to be decoded or encoded.

Value	Name	Description
0h	Reserved	This value is reserved for Monochrome Format which is not supported currently
1h	4:2:0	4:2:0 Format
2h	4:2:2	4:2:2 Format
3h	4:4:4	4:4:4 Format

28 — chroma_qp_offset_list_enabled_flag

Description
`0 – cu_chroma_qp_offset_flag is not present (default)` `1 – cu_chroma_qp_offset_flag is present.`

Programming Notes
Only for 4:4:4 can it be program to non-zero. Decoder Only

27:24 — diff_cu_chroma_qp_offset_depth

Description
Difference between the luma coding tree block size and the minimum luma coding block size of coding units that convey cu_chroma_qp_offset_flag. `Log2MinCuChromaQpOffsetSize = CtbLog2SizeY – diff_cu_chroma_qp_offset_depth` 0 to log2_diff_max_min_luma_coding_block_size default = 0. Decoder only feature

23 — Reserved

Format:	MBZ

22:20 — chroma_qp_offset_list_len_minus1

Description

		It is the index value i of the cb_qp_offset_list[i] and cr_qp_offset_list[i] syntax elements that are present in the PPS i = 0 to 5 6 to 7are invalid valid. default = 0 Decoder only feature
	19	**Reserved** Format: MBZ
	18:16	**log2_sao_offset_scale_chroma** **Description** To scale SAO offset values for chroma samples. 0 to Max(0, $BitDepth_C - 10$) default = 0 Decoder Only **Value** / **Name** 0 / 0 **[Default]**
	15	**Reserved** Format: MBZ
	14:12	**log2_sao_offset_scale_luma** **Description** To scale SAO offset values for luma samples 0 to Max(0, BitDepthC−10) Default = 0 **Value** / **Name** 0 / 0 **[Default]** **Programming Notes** Decoder only
	11:10	**MaxPCMSize** Format: U2 Specifies the largest allowed PCM coding block size. **Value** / **Name** 3 / Reserved

2	32x32
1	16x16
0	8x8

9:8 **MinPCMSize**

Format:	U2

Specifies the smallest allowed PCM coding block size.

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

7:6 **MaxTUSize**

Format:	U2

Specifies the largest allowed transform block size.

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

Programming Notes
IN VDENC mode this field should always be set to 3.

5:4 **MinTUSize**

Format:	U2

Specifies the smallest allowed transform block size.

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

Programming Notes
In VDENC mode this bit should be set to 0.

3:2 **CtbSize (LCUSize)**

Format:	U2

Specifies the coding tree block size.

HCP_PIC_STATE

Value	Name	Programming Notes
3	64x64	In VDENC, mode this should be set to 3.
2	32x32	
1	16x16	This can only be used when both picture width and height are fewer than or equal to 4222 pixels.
0	illegal/reserved	

Programming Notes
LCU is restricted based on the picture size. LCU 16x16 can only be used with picture width and height both fewer than or equal to 4222 pixels.

1:0 **MinCUSize**

Format:	U2

Specifies the smallest coding block size.

Value	Name
3	64x64
2	32x32
1	16x16
0	8x8

Programming Notes
In VDENC mode, This field should be programmed as 0.

3 | 31 **sps_range_extension_enable_flag**

Description
This flag represents both sps_extension_present_flag and sps_range_extension_flag flags in SPS. If enabled, sps_extension_present_flag=1 and sps_range_extension_flag = 0 or 1. If sps_range_extension_flag = 0, default values are set for range extension parameters. 0 – no range extension 1 – range extension Decoder only feature

30 **transform_skip_rotation_enabled_flag**

Description
0 – no rotation (default) 1 – apply rotation to the intra 4x4 residual data block coded in

HCP_PIC_STATE

		transform skip. Decoder only feature
	29	**transform_skip_context_enabled_flag** **Description** 0 – no context select for parsing sig_coeff_flag with skipped transform (default) 1 – apply context select. Decoder only feature
	28	**implicit_rdpcm_enabled_flag** **Description** 0 – the residual modification process is not used for intra blocks in the CVS (default) 1 – apply this modification process for intra blocks using a transform bypass. Decoder only feature
	27	**explicit_rdpcm_enabled_flag** **Description** 0 – the residual modification process is not used for inter blocks in the CVS (default) 1 – apply this modification process for inter blocks using a transform bypass. Decoder only feature
	26	**intra_smoothing_disabled_flag** **Description** 0 – filtering process of neighboring samples is not disabled (default) 1 – filtering process of neighboring samples is unconditionally disabled for intra prediction. Decoder only feature
	25	**persistent_rice_adaptation_enabled_flag** **Description** 0 – no previous sub-block state is used in Rice parameter derivation

HCP_PIC_STATE

	(default) 1 – use mode dependent statistics accumulated from previous sub-blocks to derive Rice parameter. Decoder only feature
24	**cabac_bypass_alignment_enabled_flag** **Description** 0 – no CABAC alignment process is used prior to bypass decoding (default) 1 – CABAC alignment process is used prior to bypass decoding coeff_sign_flag[ ] and coeff_abs_level_remaining[ ]. Decoder only feature
23	**cross_component_prediction_enabled_flag** **Description** 0 – disable cross component prediction (default) 1 – Enable cross component prediction. Only for 4:4:4 can it be program to non-zero. Decoder only feature
22:20	**Log2MaxTransformSkipSize** **Description** Equal to log2_max_transform_skip_block_size_minus2 + 2 It must less than or equal to Log2MaxTrafoSize. Decoder only feature
19	**High Precision Offsets Enable Flag** **Description** Specifies that weighted precision offset values are signaled using a bit-depth-dependent precision. 0 - Disable 1 - Enable **Programming Notes** Decoder only
18	**Reserved**

HCP_PIC_STATE

	17:8	**Reserved**
	7:4	**Reserved**
	3	**Reserved**

	2	**InsertTestFlag**
		Format: U1
		
		Value / **Name**
		0h — **[Default]**
		1h
		
		Programming Notes
		CABAC 0 Word Insertion Test Enable (Encoder Only)This bit will modify CABAC K equation so that a positive K value can be generated easily. This is done for validation purpose only. In normal usage this bit should be set to 0.
		Regular equation for generating 'K' value when CABAC 0 Word Insertion Test Enable is set to 0.
		`K = {[((96 * pic_bin_count()) - (RawMinCUBits * PicSizeInMinCUs *3) + 1023) / 1024] - bytes_in_picture} / 3`
		Modified equation when CABAC 0 Word Insertion Test Enable bit set to 1.
		`K = {[((1536 * pic_bin_count()) - (RawMinCUBits * PicSizeInMinCUs *3) + 1023) / 1024] - bytes_in_picture} / 3`
		Encoder only feature.
		
		This bit should be set to 0 in VDENC mode.
		This bit should be set to 0 in regular PAK mode.

	1	**CurPicIsI** Format: U1 Specifies that the current picture is comprised solely of I slices and that there are no P or B slices in the picture. **Value** / **Name** 0 — Current picture has at least one P or B slice **Programming Notes** This bit should be set to "0". Note: The value of "1" setting ("Current picture is comprised solely of I slices") is REMOVED. This bit is used for hardware optimization only. There is not enough information to set this bit to "1" correctly. In VDENC mode, this bit should be set to 0.

HCP_PIC_STATE

	0	**ColPicIsI** Format: U1 Specifies that the collocated picture is comprised solely of I slices and that there are no P or B slices in the picture. **Value** / **Name** 0 – Collocated picture has at least one P or B slice **Programming Notes** This bit should be set to "0". Note: The value of "1" setting ("Collocated picture is comprised solely of I slices") is REMOVED. This bit is used for hardware optimization only. There is not enough information to set this bit to "1" correctly.
4	31:28	**Reserved** Format: MBZ
	27	**CU packet structure** **Description** **Value** / **Name** / **Description** 0 – VME – CU packet structure is in output format of HW VME which is 1/2 CL per CU 1 – ExtEnc – CU packet structure is in output format of non HW VME (External Enc/SW) which is 1 CL per CU **Programming Notes** Encoder Only Value Must be zero only. ExtEnc CU packet structure is not validated. In VDENC mode, this bit should always be set to 0.
	26	**strong_intra_smoothing_enable_flag** Format: U1
	25	**transquant_bypass_enable_flag** Format: Enable **Value** / **Name** / **Description** 0 – Disable – cu_transquant_bypass is not supported 1 – Enable – cu_transquant_bypass is supported
	24	**Reserved**

HCP_PIC_STATE

	Format:	MBZ

23 amp_enabled_flag

Format:	Enable

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

Programming Notes
In VDENC mode, this bit should be set to 1.

22 transform_skip_enabled_flag

Format:	Enable

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

21 BottomField

Format:	U1

Value	Name	Description
0	Bottom Field	
1	Top Field	

Programming Notes
Must be zero for encoder only

20 FieldPic

Format:	U1

Value	Name	Description
0	Video Frame	
1	Video Field	

Programming Notes
Must be zero for encoder only.

19 weighted_pred_flag

Format:	U1

Programming Notes
In VDENC mode, this bit should be set to 1.

18 **weighted_bipred_flag**

Format:	U1

Programming Notes
In VDENC mode, this bit should be set to 1.

17 **tiles_enabled_flag**

Format:	U1

Programming Notes
This is an enable flag to enable tiling.
Current supported profile does not allow both tiles_enabled_flag andentropy_coding_sync_enabled_flag to be ON at the same time.

16 **entropy_coding_sync_enabled_flag**

Format:	U1

Not used in encoder mode

Programming Notes
Current supported profile does not allow both tiles_enabled_flag andentropy_coding_sync_enabled_flag to be ON at the same time.

15 **loop_filter_across_tiles_enabled_flag**

Format:	U1

Programming Notes
For the encoder before Gen10, this bit must be set to zero (hardware restriction). Tile support is added starting in Gen11, this field can be 0 or 1.

14 **Reserved**

Format:	MBZ

13 **sign_data_hiding_flag**

Format:	Enable

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

Programming Notes

	Currently not supported in encoder, so must be set to 0 for encoding session.
12:10	**log2_parallel_merge_level_minus2**

Format:	U3

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

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

9	**constrained_intra_pred_flag**

Format:	U1

8	**pcm_loop_filter_disable_flag**

Format:	U1

Programming Notes
In VDENC mode, this bit is always set to 1.

7:6	**diff_cu_qp_delta_depth (or named as max_dqp_depth)**

Format:	U2

Programming Notes
cu_qp_delta_enabled_flag = 1 and Max_DQP_Level = 0 or 3is supported in PAK standalone and VDenc modes.
cu_qp_delta_enabled_flag/max_dqp_depth: 1/0: has cu qp delta. (cu depth <= max_dqp_depth) will have cu qp delta coded. Only allow QP change across Slices, no change across LCU or CU.
In VDEnc mode, Max_DQP_Level can be 0 or 3.

5	**cu_qp_delta_enabled_flag**

Format:	U1

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

Programming Notes
cu_qp_delta_enabled_flag = 1 and Max_DQP_Level = 0 or 3 is supported for PAK

		standalone and VDEnc modes. In VDENC mode, this field should always be set to 1.
	4	**pcm_enabled_flag** Format: U1 **Programming Notes** In VDENC mode, this bit is always set to zero.
	3	**sample_adaptive_offset_enabled_flag** Format: U1 **Programming Notes** Restriction: HW does not support SAO filtering for LCU size 16x16 Should be set to 0 if slice_sao_luma_flag==0 and slice_sao_chroma_flag==0. It is illegal to program Slice_size_conformance ON when SAO is enabled in both first and second passes HW supports 8bit SAO and source format nv12 only. Must be off for Partial Frame Update Mode in encoder mode Restriction: HW does not support SAO filtering for LCU size 16x16 Should be set to 0 if slice_sao_luma_flag==0 and slice_sao_chroma_flag==0. HW supports both 8 and 10b SAO in single pass
	2:0	**Reserved** Format: MBZ
5	31	**Reserved** Format: MBZ
	30	**Reserved** Format: MBZ
	29:27	**bit_depth_luma_minus8** Format: U3 This specifies the number of bit allow for Luma pixels. In 8 bit mode, this must be set to 0. Encoder: Supports bit depths 8, 10 and 12 only. Encoder: Does not support 10 or 12 bit Source Pixels and 8bit PAK i.e. the source pixel depth should be less than or equal to PAK bit depth. Value / Name / Description 0 / luma_8bit / 1 / luma_9bit / Only HEVC decoder supports 9 bits luma. HEVC encoder does not supports 9 bits luma.

2	luma_10bit	
3	luma_11bit	[] Not Validated
4	luma_12bit	[] Not validated

Programming Notes
In VDENC mode, This field can only support a value of 0 or 2.

26:24 **bit_depth_chroma_minus8**

Format:	U3

This specifies the number of bit allow for Chroma pixels. In 8 bit mode, this must be set to 0. Encoder: Supports bit depths 8, 10 and 12 only. And also it must be same as Luma. Encoder: Does not support 10 or 12 bit Source Pixels and 8bit PAK. i.e. The source pixel depth should be less than or equal to the PAK bit depth.

Value	Name	Description
0	chroma_8bit	
1	chroma_9bit	Only HEVC decoder supports 9 bits chroma. HEVC encoder does not supports 9 bits chroma.
2	chroma_10bit	
3	chroma_11bit	[] Not Validated
4	chroma_12bit	[] Not validated

Programming Notes
In VDENC mode, this field should match bits 29:27 of this DW.

23:20 **pcm_sample_bit_depth_luma_minus1**

Format:	U4

Description
Encoder: Supports bit depths 8, 10 and 12 for PCM. Must be same as pixel depth. 12bit not validated
Encoder supports both 8 and 10bits starting from GEN11. Must be same for luma and Chroma Cb/Cr.

19:16 **pcm_sample_bit_depth_chroma_minus1**

Format:	U4

Description
Bit depth must me same as luma for encoder.
Encoder supports both 8 and 10bits starting from GEN11. Must be same for luma and chroma Cb and Cr.

HCP_PIC_STATE

	15:13	**max_transform_hierarchy_depth_inter(or named as tu_max_depth_inter)** Format: U3 Maximum TU split depths for inter blocks **Programming Notes** For encoder, always set to 2 to allow max 2 levels of split. For more splitting rely on CU split to match the content (Intel restriction)
	12:10	**max_transform_hierarchy_depth_intra (or named as tu_max_depth_intra)** Format: U3 Maximum TU split depth for intra blocks. **Programming Notes** For encoder, always set to 2 to allow max 2 levels of split. For more splitting, rely on CU split to match the content (Intel restriction).
	9:5	**pic_cr_qp_offset** Format: S4 Valid range -12 to +12
	4:0	**pic_cb_qp_offset** Format: S4 Valid range -12 to +12
6	31:30	**Reserved** Format: MBZ
	29	**Load Slice Pointer Flag** Format: Enable LoadBitStreamPointerPerSlice (Encoder-only) To support multiple slice picture and additional header/data insertion before and after an encoded slice. When this field is set to 0, bitstream pointer is only loaded once for the first slice of a frame. For subsequent slices in the frame, bitstream data are stitched together to form a single output data stream. When this field is set to 1, bitstream pointer is loaded for each slice of a frame. Basically bitstream data for different slices of a frame will be written to different memory locations.

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

Programming Notes
Must be zero for encoder

28:27 **Reserved**

Format:	MBZ

26 **FrameSzUnderStatusEn - FrameBitRateMinReportMask**

Format:	Enable

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

Value	Name	Description
0	Disable	[] Do not update bit 2 (Frame Bit Count Violate -- under run) of HCP_IMAGE_STATUS control register.
1	Enable	Set bit 2 (Frame Bit Count Violate -- under run) of HCP_IMAGE_STATUS control register if the total frame level bit counter is less than or equal to Frame Bit Rate Minimum limit. HW does not use this bit to set the bit in HCP_IMAGE_STATUS_CONTROL register. It's used pass the bit in HCP_IMAGE_STATUS_MASK register

Programming Notes
Encoder Only
In VDENC mode, set this bit to zero always.

25 **FrameSzOverStatusEn - FrameBitRateMaxReportMask**

Format:	Enable

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

Value	Name	Description
0	Disable	[] Do not update bit 1 of HCP_IMAGE_STATUS control register.
1	Enable	Set bit 1 of HCP_IMAGE_STATUS control register if the total frame level bit counter is greater than or equal to Frame Bit Rate Maximum limit. HW does not use this bit to set the bit in HCP_IMAGE_STATUS_CONTROL register. It's used pass the bit inHCP_IMAGE_STATUS_MASK register

Programming Notes
Encoder Only

	In VDENC mode, set this bit to zero always.
24	**LCUMaxBitStatusEn - LCUMaxSizeReportMask**

Format:	Enable

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

Value	Name	Description
0	Disable	[] Do not update bit 0 of HCP_IMAGE_STATUS control register.
1	Enable	Set bit 0 of HCP_IMAGE_STATUS control register if the total bit counter for the current LCU is greater than the LCU Conformance Max size limit. HW does not use this bit to set the bit in HCP_IMAGE_STATUS_CONTROL register.

Programming Notes
Encoder Only

23:19	**Reserved**

Format:	MBZ

18:17	**Reserved**

Format:	MBZ

16	**NonFirstPassFlag**

Format:	Enable

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

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

Programming Notes
Encoder Only

15:0	**LCU Max BitSize Allowed**

Format:	U16

Description

		This field specifies the Max LCU Bit Size allowed in according to the spec conformance. However, driver can program a different value from the spec for other encoding purpose. If LCU Limit exceeds, LCUMaxBitStatus in MMIO would be set to 1 If LCU Limit exceeds, LCUMaxBitStatus in MMIO would be set to 1 and also in CU Statistics Streamout, LCU Limit exceed field would be set to 1 at the last CU of a LCU **Programming Notes** Encoder Only For now, only the lower 18 bits are sufficient to support 8 to 12-bit 444. The extra upper bits are reserved for 16-bit pixel depth in future,
7 **Programming Notes:** Encoder Only	31	**FrameBitrateMaxUnit** Format: U1 This field is the Frame Bitrate Maximum Limit Units. 0 – Byte – 32byte unit 1 – Kilo Byte – 4kbyte unit **Programming Notes** Encoder Only
	30:14	**Reserved** Format: MBZ Reserved for future expansion of the Max Rate.
	13:0	**FrameBitRateMax** Format: U14 This field is the Frame Bitrate Maximum Limit. This field along with FrameBitrateMaxUnit determines maximum allowed bits in a frame before multi-pass gets triggered (when enabled). In other words, multi-pass is triggered when the actual frame byte count exceeds this value. [0,16383] – WhenFrameBitrateMaxUnit =0, this field is measured in unit of 32 bytes. The frame rate in bytes is range from 0-512KBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit. [0,16383] – WhenFrameBitrateMaxUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-64MBytes,

		hence the .this field is programmed from 0 to 16,384 (14-bits) unit.

8 — 31 — FrameBitrateMinUnit

Format:	U1

This field is the Frame Bitrate Minimum Limit Units.

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

Programming Notes
Encoder Only

8 — 30:14 — Reserved

Format:	MBZ

Reserved for future expansion of the Min Rate.

8 — 13:0 — FrameBitRateMin

Format:	U14

This field is the Frame Bitrate Minimum Limit. This field along with FrameBitrateMinUnit determines minimum allowed bits in a frame before multi-pass gets triggered (when enabled). In other words, multi-pass is triggered when the actual frame byte count happen to be below this value.

Value	Name	Description
[0,16383]		WhenFrameBitrateMinUnit=0, this field is measured in unit of 32 bytes.The frame rate in bytes is range from 0-512KBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.
[0,16383]		WhenFrameBitrateMinUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-64MBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.

Programming Notes
Encoder Only

9 — 31 — Reserved

Format:	MBZ

9 — 30:16 — FrameBitRateMaxDelta

Exists If:	//Always
Format:	U15

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

Value	Name	Description
0	**[Default]**	
[0,32767]		WhenFrameBitrateMaxUnit =0, this field is measured in unit of 32 bytes. The frame rate in bytes is range from 0-1024KBytes, hence the .this field is programmed from 0 to 32,767 (15-bits) unit.
[0,32767]		WhenFrameBitrateMaxUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-128MBytes, hence the .this field is programmed from 0 to 32,767 (14-bits) unit.

Programming Notes
Encoder Only

15 **Reserved**

Format:	MBZ

14:0 **FrameBitRateMinDelta**

Format:	U15

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

Value	Name	Description
0	**[Default]**	
[0,32767]		WhenFrameBitrateMinUnit =0, this field is measured in unit of 32 bytes. The frame rate in bytes is range from 0-1024KBytes, hence the .this field is programmed from 0 to 32,767 (15-bits) unit.
[0,32767]		The Programmable range is 0-128MB when FrameBitRateMinUnit is 1. WhenFrameBitrateMinUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-128MBytes, hence the .this field is programmed from 0 to 32,767 (14-bits) unit.

HCP_PIC_STATE

		Programming Notes HW requires the following condition: FrameBitRateMinDelta <= 2*FrameBitRateMinMust be true, otherwise it may cause unpredicted behavior. Encoder Only
10..11	63:0	**FrameDeltaQpMax** Format: **FrameDeltaQp** Range: [0:MAX_QP_DELTA] Frame level delta QP which should be used in case FrameSize - FrameBitRateMax in the range of ((DeltaQpMaxRange[n] * FrameBitRateMaxDelta»5)), DeltaQpMaxRange[n+1] * FrameBitRateMaxDelta»5)). **Programming Notes** If n == 7, DeltaQpMaxRange is infinity. Format: 8 bit sign-magnitude, 8bit-> Range: [0: 51] 10bit-> Range: [0:63] 12bit-> Range: [0:75] Encoder Only
12..13	63:0	**FrameDeltaQpMin** Format: **FrameDeltaQp** Range: [0:MIN_QP_DELTA] Frame level delta QP which should be used in case FrameSize - FrameBitRateMin in the range of ((DeltaQpMinRange[n] * FrameBitRateMinDelta»5)), DeltaQpMinRange[n+1] * FrameBitRateMinDelta»5)). **Programming Notes** If n == 7, DeltaQpMinRange is infinity. Format: 8 bit sign-magnitude 8bit-> Range: [-51: 0] 10bit-> Range: [-63:0] 12bit-> Range:[-75:0] Encoder Only
14..15	63:0	**FrameDeltaQpMaxRange** Format: **FrameDeltaQpRange**

HCP_PIC_STATE

		Range: [0:U8_MAX] Condition: FrameDeltaQpMaxRange[n] >= FrameDeltaQpMaxRange[n-1] This field is to calculate ranges for Frame level delta QP, specifically Frame level delta QP[n] and Frame level delta QP[n+1]. **Programming Notes** If n == 0, FrameDeltaQpMaxRange is zero. Encoder Only
16..17	63:0	**FrameDeltaQpMinRange** Format: FrameDeltaQpRange Range: [0:U8_MAX] Condition: FrameDeltaQpMinRange[n] >= FrameDeltaQpMinRange[n-1] This field is to calculate ranges for Frame level delta QP, specifically Frame level delta QP[n] and Frame level delta QP[n+1]. **Programming Notes** If n == 0, FrameDeltaQpMinRange is zero. Encoder Only
18	31:30	**MinFrameSizeUnits** Format: U2 This field is the Minimum Frame Size Units 0 – 4Kb – Minimum Frame Size is in 4Kbytes. 1 – 16Kb – Minimum Frame Size is in 16Kbytes. 2 – Compatibility Mode – [] Minimum Frame Size is in 4bytes 3 – 16 Bytes – [] Minimum Frame Size is 16 bytes. **Programming Notes** Encoder Only
	29:16	**Reserved** Format: MBZ
	15:0	**MinFrameSize**

		Default Value: 0 Format: U16 Minimum Frame Size [15:0] (in Word, 16-bit)(Encoder Only) Minimum Frame Size is specified to compensate for intel Rate Control Currently zero fill (no need to perform emulation byte insertion) is done only to the end of the CABAC_ZERO_WORD insertion (if any) at the last slice of a picture. It is needed for CBR. Intel encoder parameter. The caller should always make sure that the value, represented by Minimum Frame Size, is always less than maximum frame size FrameBitRateMax. This field is reserved in Decode mode. **Programming Notes** Programmable range is 0..(2^16-1) * 2^12 when MinFrameSizeUnits is 0. (4KB unit) Programmable range is 0..(2^16-1) * 2^14 when MinFrameSizeUnits is 1. (16KB unit) Encoder Only
32	31:30	**Reserved** Format: MBZ
	29:25	**cb_qp_offset_list[5]** Format: S4 Offsets used in the derivation of Qp'_{Cb} `Sign + 4bit (total 5-bit for each index), range +/- 12 ;` (default 0) **Programming Notes** Decoder Only
	24:20	**cb_qp_offset_list[4]** Format: S4 Offsets used in the derivation of Qp'_{Cb} `Sign + 4bit (total 5-bit for each index), range +/- 12 ;` (default 0) **Programming Notes** Decoder Only
	19:15	**cb_qp_offset_list[3]** Format: S4 Offsets used in the derivation of Qp'_{Cb}, `Sign + 4bit (total 5-bit for each index), range +/- 12 ;`

		(default 0) **Programming Notes** Decoder Only
	14:10	**cb_qp_offset_list[2]** Format: S4 Offsets used in the derivation of Qp'$_{Cb}$ `Sign + 4bit (total 5-bit for each index), range +/- 12 ;` (default 0) **Programming Notes** Decoder Only
	9:5	**cb_qp_offset_list[1]** Format: S4 Offsets used in the derivation of Qp'$_{Cb}$ `Sign + 4bit (total 5-bit for each index), range +/- 12 ;` (default 0) **Programming Notes** Decoder Only
	4:0	**cb_qp_offset_list[0]** Format: S4 Offsets used in the derivation of Qp'$_{Cb}$ `Sign + 4bit (total 5-bit for each index), range +/- 12 ;` (default 0) **Programming Notes** Decoder Only
33	31:30	**Reserved** Format: MBZ
	29:25	**cr_qp_offset_list[5]** Format: S4 Offsets used in the derivation of Qp'$_{Cb}$ `Sign + 4bit (total 5-bit for each index), range +/- 12 ;` (default 0) **Programming Notes**

		Decoder Only
	24:20	**cr_qp_offset_list[4]** Format: S4 Offsets used in the derivation of Qp'_{Cb} `Sign + 4bit (total 5-bit for each index), range +/- 12 ;` (default 0) **Programming Notes** Decoder Only
	19:15	**cr_qp_offset_list[3]** Format: S4 Offsets used in the derivation of Qp'_{Cb} `Sign + 4bit (total 5-bit for each index), range +/- 12 ;` (default 0) **Programming Notes** Decoder Only
	14:10	**cr_qp_offset_list[2]** Format: S4 Offsets used in the derivation of Qp'_{Cb} `Sign + 4bit (total 5-bit for each index), range +/- 12 ;` (default 0) **Programming Notes** Decoder Only
	9:5	**cr_qp_offset_list[1]** Format: S4 Offsets used in the derivation of Qp'_{Cb} `Sign + 4bit (total 5-bit for each index), range +/- 12 ;` (default 0) **Programming Notes** Decoder Only
	4:0	**cr_qp_offset_list[0]** Format: S4 Offsets used in the derivation of Qp'_{Cb}

		Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0) **Programming Notes** Decoder Only
34..36	31:0	**Reserved** Format: MBZ
37	31:16	**Reserved** Format: MBZ
	15:0	**RDOQIntraTUThreshold** **Description** This parameter programs the threshold for the Intra Counting. This value is programmed as a multiple of 16. When checking against the IntraTU Count, this value needs to be « 4 (multiplied by 16) and then compared against the IntraTU count to trigger the condition for Intra RDOQ disable. The IntraTU Count is accumulated in units of 4x4.

HCP_PIPE_BUF_ADDR_STATE

HCP_PIPE_BUF_ADDR_STATE

Source: VideoCS

Length Bias: 2

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

This state command provides the memory base addresses for the row store buffer and reconstructed picture output buffers required by the HCP.
This is a picture level state command and is shared by both encoding and decoding processes.

Programming Notes

All pixel surface addresses must be 4K byte aligned. There is a max of 8 Reference Picture Buffer Addresses, and all share the same third address DW in specifying 48-bit address.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: 2h HCP_PIPE_BUF_ADDR_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 60h, 72h Name:

HCP_PIPE_BUF_ADDR_STATE

1..2	63:0	**Decoded Picture** Format: SplitBaseAddress4KByteAligned Frame buffer address for the final decoded picture YUV output.
3	31:0	**Decoded Picture Memory Address Attributes** Format: MemoryAddressAttributes
4..5	63:0	**Deblocking Filter Line Buffer** Format: SplitBaseAddress64ByteAligned Base address of the filter line buffer (read/write) used by the Deblocking Filter.
6	31:0	**Deblocking Filter Line Buffer Memory Address Attributes** Format: MemoryAddressAttributes
7..8	63:0	**Deblocking Filter Tile Line Buffer** Format: SplitBaseAddress64ByteAligned Base address of the tile line buffer (read/write) used by the Deblocking Filter.
9	31:0	**Deblocking Filter Tile Line Buffer Memory Address Attributes** Format: MemoryAddressAttributes
10..11	63:0	**Deblocking Filter Tile Column Buffer** Format: SplitBaseAddress64ByteAligned Base address of the tile column buffer (read/write) used by the Deblocking Filter.
12	31:0	**Deblocking Filter Tile Column Buffer Memory Address Attributes** Format: MemoryAddressAttributes
13..14	63:0	**Metadata Line Buffer** Format: SplitBaseAddress64ByteAligned Base address for the Metadata Line buffer.
15	31:0	**Metadata Line Buffer Memory Address Attributes** Format: MemoryAddressAttributes
16..17	63:0	**Metadata Tile Line Buffer** Format: SplitBaseAddress64ByteAligned Base address for the Metadata Tile Line buffer.

HCP_PIPE_BUF_ADDR_STATE

18	31:0	**Metadata Tile Line Buffer Memory Address Attributes** Format: **MemoryAddressAttributes**
19..20	63:0	**Metadata Tile Column Buffer** Format: **SplitBaseAddress64ByteAligned** Base address for the Metadata Tile Column buffer.
21	31:0	**Metadata Tile Column Buffer Memory Address Attributes** Format: **MemoryAddressAttributes**
22..23	63:0	**SAO Line Buffer** Format: **SplitBaseAddress64ByteAligned** Base address for the SAO Line buffer.
24	31:0	**SAO Line Buffer Memory Address Attributes** Format: **MemoryAddressAttributes**
25..26	63:0	**SAO Tile Line Buffer** Format: **SplitBaseAddress64ByteAligned** Base address for the SAO Tile Line buffer.
27	31:0	**SAO Tile Line Buffer Memory Address Attributes** Format: **MemoryAddressAttributes**
28..29	63:0	**SAO Tile Column Buffer** Format: **SplitBaseAddress64ByteAligned** Base address for the SAO Tile Column buffer.
30	31:0	**SAO Tile Column Buffer Memory Address Attributes** Format: **MemoryAddressAttributes**
31..32	63:0	**Current Motion Vector Temporal Buffer** Format: **SplitBaseAddress64ByteAligned** Base address for the Current Motion Vector Temporal buffer.
33	31:0	**Current Motion Vector Temporal Buffer Memory Address Attributes**

HCP_PIPE_BUF_ADDR_STATE

		Format: **MemoryAddressAttributes**
34..35	63:0	**Reserved** Format: MBZ
36	31:0	**Reserved** Format: MBZ
37..52	511:0	**Reference Picture Base Address (RefAddr[0-7])** Format: **SplitBaseAddress64ByteAligned[8]** Base address of the reference picture buffer.
53	31:0	**Reference Picture Base Address Memory Address Attributes** Format: **MemoryAddressAttributes**
54..55	63:0	**Original Uncompressed Picture Source** Format: **SplitBaseAddress64ByteAligned** Buffer address for fetching YUV pixel data from the original uncompressed input picture for encoding. This value is only valid in **encoding** mode.
56	31:0	**Original Uncompressed Picture Source Memory Address Attributes** Format: **MemoryAddressAttributes**
57..58	63:0	**Streamout Data Destination** Exists If: //Decoder Only Format: **SplitBaseAddress64ByteAligned** Buffer address for outputting the per-block indirect data to memory when **StreamOutEnable** is set in the HCP_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. For Encoder: This surface is used to streamout CU records 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-block data.

HCP_PIPE_BUF_ADDR_STATE

59	31:0	**Streamout Data Destination Memory Address Attributes** Exists If: //Decoder Only Format: **MemoryAddressAttributes**
	31:0	**Reserved** Exists If: //Encoder Only Format: MBZ
60..61	63:0	**Decoded Picture Status/Error Buffer Base Address or Encoded slice size streamout Base Address** Format: **SplitBaseAddress64ByteAligned** **Decoder Mode :** Specifies the 64 byte aligned buffer address for writing a single status/error cache-line sized record into memory when the Pic Status/Error Report Enable is set in the HCP_PIPE_MODE_SELECT command. The pic status/error record is written by hardware after the picture is decoded. **Encoder Mode:** This specifies 64 byte aligned buffer address for writing Slice size, when slice size conformance is enabled.
62	31:0	**Decoded Picture Status/Error Buffer Base Address Memory Address Attributes** Format: **MemoryAddressAttributes**
63..64	63:0	**LCU ILDB Streamout Buffer** Format: **SplitBaseAddress64ByteAligned** Buffer address for writing ILDB parameter per LCU to memory when Deblocker Streamout Enable is set in the HCP_PIPE_MODE_SELECT Command. The ILDB MB control parameters are written by HW at the end of each reconstructed LCU. Only edge information is being streamed out.
65	31:0	**LCU ILDB Streamout Buffer Memory Address Attributes** Format: **MemoryAddressAttributes**
66..81	511:0	**Collocated Motion Vector Temporal Buffer[0-7]** Format: **SplitBaseAddress64ByteAligned[8]** Base address for the Collocated Motion Vector Temporal buffer.
82	31:0	**Collocated Motion Vector Temporal Buffer[0-7] Memory Address Attributes** Format: **MemoryAddressAttributes**

HCP_PIPE_BUF_ADDR_STATE

83..84	63:0	**VP9 Probability Buffer Read/Write** Format: **SplitBaseAddress64ByteAligned** Specifies the 64 byte aligned buffer address for VP9 Probability Buffer. Hardware reads in the probability for decode and write out the modified probability for future frames. Driver needs to program the Initial VP9 Probability for decoding the current frame. For Key Frame, it should contain the default Key Frame Probability. For non-Key Frame, it could be a default (non-Key) or one of the 8 Reference Buffers Probability. Driver must provide a valid Initial VP9 Probability buffer.
85	31:0	**VP9 Probability Buffer Read/Write Memory Address Attributes** Format: **MemoryAddressAttributes**
86..87	63:0	**VP9 Segment ID Buffer Read/Write** Specifies the 64 byte aligned buffer address for VP9 SegmentID buffer. This should contain the writeout SegmentID from previous frame and will be used to predict SegmentID for the current frame. Hardware will write out SegmentID of the current frame in the same address for the next frame.
88	31:0	**VP9 Segment ID buffer Read/Write Memory Address Attributes** Format: **MemoryAddressAttributes**
89..90	63:0	**VP9 HVD Line Rowstore Buffer Read/Write** Format: **SplitBaseAddress64ByteAligned** Specifies the 64 byte aligned buffer address for HVD Tile Rowstore Buffer (bitstream decoder).
91	31:0	**VP9 HVD Line Rowstore buffer Read/Write Memory Address Attributes** Format: **MemoryAddressAttributes**
92..93	63:0	**VP9 HVD Tile Rowstore Buffer Read/Write** Format: **SplitBaseAddress64ByteAligned**
94	31:0	**VP9 HVD Tile Rowstore buffer Read/Write Memory Address Attributes** Format: **MemoryAddressAttributes**
95..96	63:0	**SAO Rowstore Buffer Base Address**

HCP_PIPE_BUF_ADDR_STATE

		Specifies the 64 byte aligned buffer address for Rowstoring of SAO parameters in encoder mode
97	31:0	**SAO Rowstore Buffer Read/Write Memory Address Attributes** Format: **MemoryAddressAttributes**
104..105	63:0	**HCP Scalability Slice State Buffer Base Address** Format: **SplitBaseAddress64ByteAligned** Specifies the 64 byte aligned buffer address for storing slice state information on HEVC/VP9 Scalable mode for decode. This is needed since CABAC and BE pass will be separated and BE pass needs to have slice state information as well. This buffer is only used in HEVC Scalable Decode Mode Only (Virtual Tile on both CABAC and Recon Pass)
106	31:0	**HCP Scalability Slice State Buffer (attributes) Read/Write** Format: **MemoryAddressAttributes**
107..108	63:0	**HCP Scalability CABAC Decoded Syntax Elements Buffer Base Address** Format: **SplitBaseAddress64ByteAligned** Specifies the 64 byte aligned buffer address for storing CABAC Decoded Syntax Element on HEVC/VP9 Scalable mode for decode
109	31:0	**HCP Scalability CABAC Decoded Syntax Elements Buffer (attributes) Read/Write** Format: **MemoryAddressAttributes**
110..111	63:0	**Motion Vector Upper Right Column Store Buffer Base Address** Format: **SplitBaseAddress64ByteAligned** Specifies the 64 byte aligned buffer address for storing upper right Motion Vector on HEVC/VP9 Scalable mode for decode in BE pass. This buffer is used to pass data across pipes for multiple pipe mode. This buffer is only used on HEVC Scalable Decode Only (Virtual Tile on both CABAC and Recon pass)
112	31:0	**Motion Vector Upper Right Column Store Buffer (attributes) Read/Write**

HCP_PIPE_BUF_ADDR_STATE

		Format: **MemoryAddressAttributes**
113..114	63:0	**Intra Prediction Upper Right Column Store Buffer Base Address** Format: **SplitBaseAddress64ByteAligned** Specifies the 64 byte aligned buffer address for storing upper right Intra Prediction Pixel on HEVC/VP9 Scalable mode for decode in BE pass. This buffer is used to pass data across pipes for multiple pipe mode.
115	31:0	**Intra Prediction Upper Right Column Store Buffer (attributes) Read/Write** Format: **MemoryAddressAttributes**
116..117	63:0	**Intra Prediction Left Recon Column Store Buffer Base Address** Format: **SplitBaseAddress64ByteAligned** Specifies the 64 byte aligned buffer address for storing left column Intra Prediction Pixel on HEVC/VP9 Scalable mode for decode in BE pass. This buffer is used to pass data across pipes for multiple pipe mode.
118	31:0	**Intra Prediction Left Recon Column Store Buffer (attributes) Read/Write** Format: **MemoryAddressAttributes**
119..120	63:0	**HCP Scalability CABAC Decoded Syntax Elements Buffer Max Address** Format: **SplitBaseAddress64ByteAligned** Specifies the 64 byte aligned maximum address for the HCP Scalability CABAC Decoded Syntax Elements Buffer. This address shall either be 0 or larger than HCP Scalability CABAC Decoded Syntax Elements Buffer Base Address. If this address is 0, the upper bound is considered disable and HW will NOT check for upper bound. Hardware shall only write to memory address less than this address (unless address is 0 which is disabled). Hardware will not write to memory address larger than or equal to address (unless address is 0 which is disabled)

HCP_PIPE_MODE_SELECT

HCP_PIPE_MODE_SELECT

Source: VideoCS

Length Bias: 2

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

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

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

HCP_PIPE_MODE_SELECT

	23	**Reserved**
	22:18	**Reserved**
		Format: MBZ
	17	**RESERVED**
		Format: MBZ
	16:15	**Pipe working Mode**

This programs the working mode for HCP pipe.

Value	Name	Description
00b	Legacy decoder/encoder mode (Single pipe)	This is for single pipe mode non-scalable mode. It is used by both decoder and encoder.
01b	CABAC FE only decode mode (Single CABAC pipe)	This is for the single CABAC FE only in decoder mode. This will be only run CABAC and streamout syntax element.
10b	Decoder BE only or Encoder mode (Scalable Multi-pipe)	This is for multiple-pipe scalable mode. In decoder, it is only on BE reconstruction. In encoder, it is for PAK.
11b	Reserved	

	14:13	**Multi-Engine Mode**

This indicates the current pipe is in single pipe mode or if in scalable mode is in left/right/middle pipe in multi-engine mode.

Value	Name	Description
00b	Single Engine Mode or CABAC FE only decode mode	This is for single engine mode (legacy) OR CABAC FE only decode mode
01b	Pipe is the left engine in a Multi-engine mode	Current pipe is the most left engine while running in scalable multi-engine mode
10b	Pipe is the right engine in a Multi-engine mode	Current pipe is the most right engine while running in scalable multi-engine mode
11b	Pipe is one of the middle engine in a Multi-engine mode	Current pipe is in one of the middle engine while running in scalable multi-engine mode

	12	**Reserved**
		Format: MBZ
	11	**Reserved**

HCP_PIPE_MODE_SELECT

	Format:	MBZ

10	**Reserved**

8	**Reserved**

7:5 Codec Standard Select

Value	Name
0	HEVC
1	VP9

4 Reserved

This bit is reserved since it is used. Making sure there is no overlap between the two commands.

3 Pic Status/Error Report Enable

Format:	Enable

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

2 PAK Pipeline Streamout Enable

Format:	Enable

Pipeline Streamout Enable is only defined for encode. It is ignored for decode.

Value	Name
0	Disable pipeline states and parameters streamout
1	Enable pipeline states and parameters streamout

Programming Notes
In VDENC mode, this field should be set to 1 always.

1 Deblocker Streamout Enable

Format:	Enable

Deblocker Streamout Enable not currently supported for Encode or Decode

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

HCP_PIPE_MODE_SELECT

	0	**Codec Select** Format: U1 **Value** / **Name** 0 – Decode 1 – Encode
2	31:0	**Media Soft-Reset Counter (per 1000 clocks)** Format: U32 In decoder modes, this counter value specifies the number of clocks (per 1000) of GAC inactivity before a media soft-reset is applied to the HCP. If counter value is set to 0, the media soft-reset feature is disabled and no reset will occur. In encoder modes, this counter must be set to 0 to disable media soft reset. This feature is not supported for the encoder. **Value** / **Name** 0 – Disable
3	31:0	**Pic Status/Error Report ID** Format: U32 The Pic Status/Error Report ID is a unique 32-bit unsigned integer assigned to each picture status/error output. Must be zero for encoder mode. **Value** / **Name** / **Description** 0 – 32-bit unsigned – Unique ID Number 1 – Reserved **Programming Notes** Software must program different Status/Error Buffer addresses between pictures; otherwise the hardware might overwrite previously written data.
4	31:0	**Reserved** Format: MBZ
5	31:0	**Reserved** Format: MBZ

HCP_QM_STATE

HCP_QM_STATE

Source: VideoCS

Length Bias: 2

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

The HCP_QM_STATE command loads the custom HEVC quantization tables into local RAM and may be issued up to 20 times: 3x Color Component plus 2x intra/inter plus 4x SizeID minus 4 for the 32x32 chroma components.
When the scaling_list_enable_flag is set to disable, the scaling matrix is still sent to the decoder, and with all entries programmed to the same value = 16.
This is a picture level state command and is issued in both encoding and decoding processes.

Dwords 2-17 form a table for the DCT coefficients, 4 8-bit coefficients/DWord.

- Size 4x4 for SizeID0, DWords 2-5.
- Size 8x8 for SizeID1/2/3, DWords 2-17.

SizeID 0 (Table 4-10)

4x4	[31:24]	[23:16]	[15:8]	[7:0]
DWord 2	AC(0,3)	AC(0,2)	AC(0,1)	DC
DWord 3	AC(1,3)	AC(1,2)	AC(1,1)	AC(1,0)
DWord 4	AC(2,3)	AC(2,2)	AC(2,1)	AC(2,0)
DWord 5	AC(3,3)	AC(3,2)	AC(3,1)	AC(3,0)

SizeID 1, 2, 3 (Table 4-11)

8x8	[31:24]	[23:16]	[15:8]	[7:0]	[31:24]	[23:16]	[15:8]	[7:0]
DWord 3,2	AC(0,7)	AC(0,6)	AC(0,5)	AC(0,4)	AC(0,3)	AC(0,2)	AC(0,1)	DC
DWord 5,4	AC(1,7)	AC(1,6)	AC(1,5)	AC(1,4)	AC(1,3)	AC(1,2)	AC(1,1)	AC(1,0)
DWord 7,6	AC(2,7)	AC(2,6)	AC(2,5)	AC(2,4)	AC(2,3)	AC(2,2)	AC(2,1)	AC(2,0)
DWord 9,8	AC(3,7)	AC(3,6)	AC(3,5)	AC(3,4)	AC(3,3)	AC(3,2)	AC(3,1)	AC(3,0)
DWord 11,10	AC(4,7)	AC(4,6)	AC(4,5)	AC(4,4)	AC(4,3)	AC(4,2)	AC(4,1)	AC(4,0)
DWord 13,12	AC(5,7)	AC(5,6)	AC(5,5)	AC(5,4)	AC(5,3)	AC(5,2)	AC(5,1)	AC(5,0)
DWord 15,14	AC(6,7)	AC(6,6)	AC(6,5)	AC(6,4)	AC(6,3)	AC(6,2)	AC(6,1)	AC(6,0)
DWord 17,16	AC(7,7)	AC(7,6)	AC(7,5)	AC(7,4)	AC(7,3)	AC(7,2)	AC(7,1)	AC(7,0)

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE

HCP_QM_STATE

		Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: 4h HCP_QM_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 10h — Name:
1	31	**Reserved** Format: MBZ
	30:13	**Reserved** Format: MBZ
	12:5	**DC Coefficient** Format: U8 Specifies the 8-bit DC coefficient for SizeID 2 and 3. **Programming Notes** The DC Coefficient must be set to zero for SizeID 0 and 1. The DC Coefficient must be set to scaling_list_dc_coef_minus8 + 8 for SizeID 2 and 3.
	4:3	**Color Component** Format: U2 Encoder: When RDOQ is enabled, scaling list for all 3 color components must be same. So this field is set to always 0. Value: 0 — Name: Luma

1	Chroma Cb
2	Chroma Cr
3	Reserved

2:1 SizeID

Format:	U2

Value	Name	Description
0	4x4	
1	8x8	
2	16x16	
3	32x32	(Illegal Value for Colour Component Chroma Cr and Cb.)

0 Prediction Type

Format:	U1

Value	Name
0	Intra
1	Inter

2..17 511:0 QuantizerMatrix

HCP_REF_IDX_STATE

HCP_REF_IDX_STATE

Source: VideoCS

Length Bias: 2

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

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

Unlike AVC, HEVC allows 16 reference idx entries in each of the L0 and L1 list for a progressive picture. Hence, a max total 32 reference idx in both lists together. The same when the picture is a field picture. Regardless the number of reference idx entries, there are only max 8 reference pictures exist at any one time. Multiple reference idx can point to the same reference picture and can optionally pic a top or bottom field, or frame.

For P-Slice, this command is issued only once, representing L0 list. For B-Slice, this command can be issued up to two times, one for L0 list and one for L1 list.

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

HCP_REF_IDX_STATE

		10h
1	31:5	**Reserved** Format: MBZ
	4:1	**num_ref_idx_l[RefPicListNum]_active_minus1** Format: U4 num_ref_idx_l[*RefPicListNum*]_active_minus1 **Value** / **Name** [0-14]
	0	**RefPicListNum** Format: U1 **Value** / **Name** 0 – Reference Picture List 0 1 – Reference Picture List 1
2..17	511:0	**Entries** Format: **HCP_REF_LIST_ENTRY[16]**

HCP_SLICE_STATE

HCP_SLICE_STATE

Source: VideoCS

Length Bias: 2

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

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: 14h HCP_SLICE_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value / Name 9h 7h
1	31:26	**Reserved** Format: MBZ
	25:16	**SliceStartCtbY or (slice_start_lcu_y encoder)**

		Format: U10 Specifies the starting row address of the first coding tree block in the current slice.
	15:10	**Reserved** Format: MBZ
	9:0	**SliceStartCtbX or (slice_start_lcu_x encoder)** Format: U10 Specifies the starting column address of the first coding tree block in the current slice. **Programming Notes** In VDENC mode, this bit should be programmed as zero
2	31:27	**Reserved** Format: MBZ
	26:16	**NextSliceStartCtbY or (next_slice_start_lcu_y encoder)** Format: U11 Specifies the starting row address of the first coding tree block in the next slice. Must be set to zero when the current slice is the last slice of a picture. For the single slice per frame case, the only slice is also the last slice, so this parameter should be set to a number larger than the frame height (at least +1).
	15	**Reserved** Format: MBZ
	14:10	**Reserved** Format: MBZ
	9:0	**NextSliceStartCtbX or (next_slice_start_lcu_x encoder)** Format: U10 Specifies the starting column address of the first coding tree block in the next slice. Must be set to zero when the current slice is the last slice of a picture. For the single slice per frame case, the only slice is also the last slice, so this parameter should be set to a number larger than the frame width (at least +1). **Programming Notes**

		In VDENC mode, this field should always be programmed as zero
3	31:26	**Reserved** Format: MBZ
	25	**LastSliceOfTileColumn** Format: U1 Indicates Last Slice of a Tile Column
	24	**LastSliceOfTile** Format: U1 Indicates last slice of a Tile
	23	**cu_chroma_qp_offset_enabled_flag** Format: Enable This specifies that the cu_chroma_qp_offset_flag may be present in the transform unit syntax. chroma_qp_offset_list_enabled_flag equal to 0 specifies that the cu_chroma_qp_offset_flag is not present in the transform unit syntax. When ChromaArrayType is equal to 0, it is a requirement of bitstream conformance that the value of chroma_qp_offset_list_enabled_flag shall be equal to 0. **Programming Notes** Decoder only feature.
	22	**Reserved**
	21:17	**slice_cr_qp_offset** Format: S4 For deblocking purpose, the pic and slice level cr qp offset must be provided separately. PAK needs to perform final_chroma_cr_qp_offset = pic_cr_qp_offset + slice_cr_qp_offset.

Value	Name
14h	-12
15h	-11
16h	-10
17h	-9
18h	-8
19h	-7

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

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

16:12 **slice_cb_qp_offset**

Format:	S4

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

Value	Name
14h	-12
15h	-11
16h	-10
17h	-9
18h	-8
19h	-7
1Ah	-6

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

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

11:6 **SliceQp**

Format:	U6

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

Programming Notes
In 12 bit, this needs to go from -24 to 51

5 **slice_temporal_mvp_enable_flag**

Format:	U1

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

4 **dependent_slice_flag**

Format:	U1

Decoder only.

HCP_SLICE_STATE

	3	**SliceQp Sign Flag** Format: U1 This specifies the sign bit of SliceQp. This is added for HEVC 10 bit. For 8 bit, SliceQp goes from 0 to 51 so this bit should be zero. In 10 bit, SliceQp goes from -12 to 51 and this bit can be set for negative value.
	2	**LastSliceofPic** Format: U1 This indicates the current slice is the very last slice of the current picture Value: 0 — Not the last slice of the picture Value: 1 — Last slice of the picture
	1:0	**slice_type** Format: U2 Value: 0 — B-slice Value: 1 — P-slice Value: 2 — I-slice Value: 3 — Illegal/Reserved **Programming Notes** In VDENC mode, for HEVC standard this field can be 0 or 2 only.
4	31:29	**Reserved** Format: MBZ
	28:26	**CollocatedRefIDX** Format: U3 Collocated Motion Vector Temporal Buffer Index. **Programming Notes** In VDENC mode, the valid values are 0,1,2.
	25:23	**MaxMergeIDX** Format: U3 MaxNumMergeCand = 5 - five_minus_max_num_merge_cand -1. Value: 0 — Name: 0

	1	1
	2	2
	3	3
	4	4

Programming Notes

The valid value is from 0 to 4
(MaxNumMergeCand = 5 - five_minus_max_num_merge_cand -1)

In VDENC mode, this field should be programmed as 4.

22	**cabac_init_flag**	
	Format:	U1

Programming Notes

In VDEnc Mode, this value should be the same accross all the slices within frame.

21:19	**luma_log2_weight_denom**	
	Format:	U3
18:16	**ChromaLog2WeightDenom**	
	Format:	U3
15	**collocated_from_l0_flag**	
	Format:	U1
14	**isLowDelay**	
	Format:	U1

If the POCs of all pictures in both lists are less than the current POC, then set to one, else set to zero.

Programming Notes

In VDENC mode, this bit should be set to 1.

13	**mvd_l1_zero_flag**	
	Format:	U1

Decoder only.

12	**slice_sao_luma_flag**	
	Format:	U1

Programming Notes

Note: For encoder, all Slices must have same setting within a picture

11	**slice_sao_chroma_flag**	
	Format:	U1

HCP_SLICE_STATE

		Programming Notes Note: For encoder, all Slices must have same setting within a picture
	10	**slice_loop_filter_across_slices_enabled_flag** Format: U1 **Programming Notes** In VDENC mode, this bit should be set to 1 always. No encoder restriction going forward from Gen11
	9	**Reserved** Format: MBZ
	8:5	**slice_beta_offset_div2 or (final Beta_Offset_div2 Encoder)** Format: S3 Deblocking filter beta offset. Specified in 2's comp. **Value** [1101b,0011b] — **Name** [-3,3] **Programming Notes** Note: In VDEnc mode, the value should be the same across all the slices within frame. Valid only in encoder mode
	4:1	**slice_tc_offset_div2 or (final tc_offset_div2 Encoder)** Format: S3 Deblocking filter tc offset. Specified in 2's comp. **Value** [1101b,0011b] — **Name** [-3,3] **Programming Notes** Note: In VDEnc mode, the value should be the same across all the slices within frame. Valid only in encoder mode
	0	**slice_header_disable_deblocking_filter_flag** Format: U1 **Programming Notes** In VDENC mode, set this bit to zero always.
5	31:16	**Reserved** Format: MBZ
	15:0	**SliceHeaderLength**

		Format: U16
		Decoder only. Specifies the length in bytes of the slice header including the start code. The starting byte of the slice header in the bit stream buffer is indicated by the Indirect Data Start Address in the HCP_BSD_OBJECT command. The ending byte of the slice header in the same bit stream buffer is indicated by the last byte prior to the slice data (CABAC).
6	31:30	**Reserved** Format: MBZ
	29:26	**RoundInter** Format: U4 In VDENC mode, this field is ignored.

Value	Name
0h	+1/32
1h	+2/32
2h	+3/32
3h	+4/32
4h	+5/32 **[Default]**
5h	+6/32
6h	+7/32
7h	+8/32
8h	+9/32
9h	+10/32
Ah	+11/32
Bh	+12/32
Ch	+13/32
Dh	+14/32
Eh	+15/32
Fh	+16/32

Programming Notes
Encoder only feature

	25:24	**Reserved** Format: MBZ

	23:20	**RoundIntra** Format: U4 In VDENC mode, this field is ignored. **Value** / **Name** 0h: +1/32 1h: +2/32 2h: +3/32 3h: +4/32 4h: +5/32 **[Default]** 5h: +6/32 6h: +7/32 7h: +8/32 8h: +9/32 9h: +10/32 Ah: +11/32 Bh: +12/32 Ch: +13/32 Dh: +14/32 Eh: +15/32 Fh: +16/32 **Programming Notes** Encoder only feature
	19:0	**Reserved** Format: MBZ
7	31:11	**Reserved** Format: MBZ
	10	**Header Insertion Enable** Format: U1 Must be followed by the PAK Insertion Object Command to perform the actual insertion. **Value** / **Name** / **Description** 0 / / No header insertion into the output bitstream buffer, before the current slice

		encoded bits.
1		Header insertion into the output bitstream buffer is present, and is before the current slice encoded bits.

Programming Notes
Must be always enabled. Encoder Only feature
In VDENC mode, this bit should be set to 1.

9 **SliceData Enable**

Format:	U1

Must always be enabled.
Encoder only feature.

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

Programming Notes
In VDENC mode, this bit should be set to 1.

8 **Tail Insertion Enable**

Format:	U1

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

Value	Name	Description
0		No tail insertion into the output bitstream buffer, after the current slice encoded bits.
1		Tail insertion into the output bitstream buffer is present, and is after the current slice encoded bits. Tail insertion is only possible at the end of frame but not in the middle (say slice end)

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

7:3 **Reserved**

Format:	MBZ

	2	**EmulationByteSliceInsertEnable** Format: U1 To have PAK outputting SODB or EBSP to the output bitstream buffer. 0 – outputting RBSP 1 – outputting EBSP **Programming Notes** Encoder Only feature In VDENC mode this bit should be set to 1.
	1	**CabacZeroWordInsertionEnable** Format: U1 To pad the end of a SliceLayer RBSP to meet the encoded size requirement. 0 – No Cabac_Zero_Word Insertion. 1 – Allow internal Cabac_Zero_Word generation and append to the end of RBSP (effectively can be used as an indicator for last slice of a picture, if the assumption is only the last slice of a picture needs to insert CABAC_ZERO_WORDs). **Programming Notes** Encoder Only feature
	0	**Reserved** Format: MBZ
8	31:29	**Reserved** Format: MBZ
	28:6	**Indirect PAK-BSE Data Start Offset (Write)** Format: U23 This field specifies the memory starting address (offset) to write out the compressed bitstream data from the BSE processing. This pointer is relative to the HCP PAK-BSE Object Base Address. It is a cacheline-aligned address for the HEVC bitstream data.

For Write, there is no need to have a data length field. It is assumed the global memory upper bound check specified in the IND_OBJ_BASE_ADDRESS command (Indirect PAK-BSE Object Access Upper Bound) will take care of any illegal write access.

Value	Name
[0,524288]	

Programming Notes
Must be zero.
Encoder Only feature
In VDENC mode, this field should be zero.

5:0 **Reserved**

Format:	MBZ

9 31 **Force SAO parameters to zero**

Description
1->Force SAO parameters to zero 0->normal compute This bit does not change SAO compute but forces SAO parameters to zero at the end. Default should be 0. All slices must be programmed to same value within a frame. This bit is used for rate control purpose on last pass along with delta QP to decrease bitstream size. All previous passes still use deltaQP to decrease/increase bitstream size. HW might not hit the last pass.

Programming Notes
Encoder Only feature

30:16 **Reserved**

15:0 **Transformskip_lambda**

Lambda value used in transform skip calculation

Programming Notes
Encoder Only feature

10 31:24 **Transformskip_numnonzerocoeffs_factor1**

Multiplying factor with number of non-zero coefficients for non-Skip pipe calculation

HCP_SLICE_STATE		
		Programming Notes
		Encoder Only feature
	23:16	**Transformskip_numzerocoeffs_factor1**
		Multiplying factor with number of zero coefficients for non-Skip pipe calculation
		Programming Notes
		Encoder Only feature
	15:8	**Transformskip_numnonzerocoeffs_factor0**
		Multiplying factor with number of non-zero coefficients for skip pipe calculation
		Programming Notes
		Encoder Only
	7:0	**Transformskip_numzerocoeffs_factor0**
		Multiplying factor with number of zero coefficients for skip pipe calculation
		Programming Notes
		Encoder Only feature

HCP_SURFACE_STATE

HCP_SURFACE_STATE

Source: VideoCS

Length Bias: 2

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

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

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

HCP_SURFACE_STATE

		Format: U4

Value	Name	Description
0h	HEVC: For current decoded Picture	8-bit uncompressed data
1h	Source Input Picture (encoder)	8-bit uncompressed data
2h	Prev Reference Picture	(VP9 only) Previous Reference
3h	Golden Reference Picture	(VP9 only) Golden Reference
4h	AltRef Reference Picture	(VP9 only) AltRef Reference

27:17 **Reserved**

Format:	MBZ

16:0 **Surface Pitch Minus1**

Format:	U17-1

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

Programming Notes
For TileYF and TileYS surfaces, the range is dependent on the Cu parameter (refer to Memory Data Formats section for the definition of the Cu parameter depending on the case). The range in bytes is [2^{Cu}-1, 131071] -> [(2^{Cu})B, 128KB] = [1 tile, 128KB/($2^{Cu)\ tiles}$] The field specifies the surface pitch in (#Bytes - 1)
For tiled surfaces, the pitch must be a multiple of the tile width (i.e.128 bytes aligned). If Half Pitch for Chroma is set, this field must be a multiple of two tile widths for tiled surfaces, or a multiple of 2 bytes for linear surfaces. For Y-tiled surfaces: Range = [127, 131071] to [128B,128KB] = [1 tile, 1024 tiles]

2 31:27 **Surface Format**

Format:	U5

Specifies the format of the surface.

Value	Name	Description
0h	YUY2 format	
1h	RGB_8 format	
2h	AYUV4444 format	
4h	PLANAR_420_8	
5h	YCRCB_SwapY format	
6h	YCRCB_SwapUV format	
7h	YCRCB_SwapUVY format	
8h	Y216/Y210 format	[] Same value is used to represent Y216 and Y210

9h	RGB_10 format	
Ah	Y410 format	
Bh	NV21 Planar_420_8 Format	
Ch	Y416 format	
Dh	P010	
11h	Y216Variant	Y216Variant is the modified Y210/Y216 format, 8 bit planar 422 with MSB bytes packed together and LSB bytes at an offset in the X-direction where the x-offset is 32-bit aligned. The chroma is UV interleaved with identical MSB and LSB split as luma and is at an offset in the Y-direction (similar to NV12) but is the same height as the luma.
12h	Y416Variant	Y416Variant is the modified Y410/Y412/Y416 format,8 bit planar 444 with MSB bytes packed together and LSB bytes at an offset in the X-direction where the x-offset is 32-bit aligned. The U channel is below the luma, has identical MSB and LSB split as luma and is at an offset in the Y-direction (similar to NV12) but is the same height as the luma The V channel is below the U, has identical MSB and LSB split as luma and is at an offset in the Y-direction (similar to NV12) but is the same height as the luma.
13h	YUY2Variant	YUY2Variant is the modified YUY2 format, 8 bit planar 422. The chroma is UV interleaved and is at an offset in the Y-direction (similar to NV12) but is the same height as the luma.
14h	AYUV4444Variant	AYUV4444Variant is the modified AYUV4444 format, 8 bit planar 444 format. The U channel is below the luma and is at an offset in the Y-direction (similar to NV12) but is the same height as the luma. The V channel is below the and is at an offset in the Y-direction (similar to NV12) but is the same height as the luma.
15h-1Fh	Reserved	

Programming Notes

Programming restriction on HEVC decoder:

1. If both luma_bitdepth_minus8 and chroma_bitdepth_minus 8 are both 0 (8 bits for both luma/chroma), this should be programmed to PLANAR_420_8
2. If either luma_bitdepth_minus8 or chroma_bitdepth_minus 8 is non-zero (9 or 10 bits for either or both luma/chroma), this should be programmed to P010.

26 **Reserved**

Format:	MBZ

	25	**Reserved** Format: MBZ
	24:15	**Reserved** Format: MBZ
	14:0	**Y Offset for U(Cb) in pixel** Format: U15_Pixel_Row_Offset This field specifies the vertical offset in rows from the **Surface Base Address** to the start (origin) of the U(Cb) plane or the interleaved UV plane if **Interleave Chroma** is enabled. This field is only used for PLANAR surface formats. **Programming Notes** • For PLANAR_420 surface formats, the alignment of this field follows the tile mode described in bits 14:13 of the **Memory Address Attributes** table. • TileY (legacy 4k) - 8 pixel aligned • TileYF (New 4k) - 64 pixel aligned • TileYS (64k) - 256 pixel aligned
3	15:0	**Default Alpha Value**

HCP_TILE_CODING

HCP_TILE_CODING

Source: BSpec

Length Bias: 1

Programming Notes
This command is used for both HEVC and VP9 codecs

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: Opcode
	28:27	**Pipeline Type** Default Value: 2h Format: Opcode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: Opcode
	22:16	**Media Instruction Command** Default Value: 15h HCP_TILE_CODING Format: Opcode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: U12 Value: Dh — Name: **[Default]**
1	31:16	**Num of Tile columns in a Frame** Description: Specify the total number of Tile columns in a frame. This field is not used by HW. Restriction : This field should be same as "Number of Active BE Pipes" defined in this register for the following modes: (1) HEVC/VP9 virtual tile decode mode

		(2) HEVC/VP9 PAK (encoder) mode
	15:10	**Reserved MBZ**
	9:8	**Reserved**
	7:0	**Number of Active BE Pipes**

Description

Indicates the number of active, consecutive positioned Scalable VDBOXs to be used for the current frame decoding or encoding.
BE Pipe partitioning, SW must guarantee the minimum width is at least two full LCUs for each tiles

This field in general should be smaller or equal to Num of Tile columns in a Frame.
This field is ignored by HW

This field is not used by HW

Value	Comment
0	ignored
1	ignored
2	Supported by Encoder / Decoder.
3	Supported only by Decoder.
4	Supported only by Encoder

2	31	**IsLastTileOfColumn** Format: U1 Indicates if current Tile is last tile of a Column
	30	**Reserved** Format: MBZ
	29:26	**Reserved** Format: MBZ
	25:16	**Tile Row Position** Format: U10 Ctb row position of tile

HCP_TILE_CODING

		For VP9: In units of SB64x64
	15:11	**Reserved** | | | |
	10	**Non First Pass Tile** | | | | Indicates Non first past Tile when Tile Replay Mode is enabled
	9:0	**Tile Column Position** | | | | Format: | U10 | Ctb column position of tile For VP9: In units of SB64x64
3	31	**Reserved** | | | | Format: | MBZ |
	30:27	**Reserved** | | | | Format: | MBZ |
	26:16	**TileWidthInMinCbMinus1** | | | | **Description** Specifies Tile width in units of minimum coding block size. The minimal width per tile is at least two full LCUs. In HEVC Encoder mode, the following restrictions apply. Last LCU at frame's right edge must align to CU boundary. This applies to all size of LCUs: 16x16, 32x32, 64x64. Kernel does not count/include CUs outside of the picture in PakObj/CU_record respectively. For VP9: In units of 8x8
	15:11	**Reserved** | | | | Format: | MBZ |
	10:0	**TileHeightInMinCbMinus1** | | | | **Description**

		Specifies Tile Height in units of minimum coding block size In HEVC Encoder mode, the following restrictions apply. Last LCU at frame's bottom edge must align to CU boundary. This applies to all size of LCUs: 16x16, 32x32, 64x64. Kernel does not count/include CUs outside of the picture in PakObj/CU_record respectively. For VP9: In units of 8x8
4	31:6	**Bitstream Byte Offset** Offset on top of base address from where the encoded bitstreamshould be written out for this tile In scalability mode: this offset is valid for every tile and it must be zero for the first tile in a frame. Non scalability mode: not valid
	5:1	**Reserved** Format: MBZ
	0	**Reserved** Format: MBZ
5	31:6	**PAK Frame Statistics Offset** The frame statistics (SSE, RhoDomain and LCU stats) will be reported per Tile. Valid only in scalability mode
	5:0	**Reserved** Format: MBZ
6	31:6	**CU Level Streamout Offset** CU level statistics (see details in streamout section) per tile will be streamed out starting from this offset address This offset is valid for every Tile in scalability mode only.
	5:0	**Reserved** Format: MBZ
7	31:6	**Slice Size Streamout Offset** Size of every slice within this tile will be streamed out startingfrom thisoffset address.

		This offset is valid for every Tile in scalability mode only.
	5:0	**Reserved** Format: MBZ
8	31:6	**CU record offset** Offset address for CU record for this tile This offset is valid for every Tile in scalability mode only.
	5:0	**Reserved** Format: MBZ
9	31:6	**SSE RowStore offset** SSE(Sum Square Error) statistics per tile will be written out at this address This offset is valid for every Tile in scalability mode.
	5:0	**Reserved** Format: MBZ
10	31:6	**SAO RowStore offset** SAO Rowstore offset for this tile. This offset is valid for every Tile in scalability mode only.
	5:0	**Reserved** Format: MBZ
11	31:6	**Tile Size StreamOut Offset** Tile Size will be written out at this offset This offset is valid for every Tile in scalability mode only.
	5:0	**Reserved** Format: MBZ
12	31:6	**VP9 Probability Counter Streamout Offset**

HCP_TILE_CODING

		Probability counters will be written out starting from this offset address This offset is valid for every Tile in scalability mode only.
	5:0	**Reserved** Format: MBZ
13..14	63:0	**HCP Scalability Synchronize Buffer - Base Address** Format: **SplitBaseAddress64ByteAligned** Specifies the 64 byte aligned buffer address used for data synchronization between neighboring pipes in scalable modes. The buffer will be written and read (as a flush mechanism) by hardware to guarantee data made it to memory before neighboring pipe can read the data. Hardware will also write the current row (in LCU) number to indicate which the current processing rows. **Programming Notes** This minimal buffer size (in CLs) should be set to the number of scalable pipes used by this workload. This data should be not be cached.
15	31:0	**HCP Scalability Synchronize Buffer - Attributes** Format: **MemoryAddressAttributes**
16	31:0	**Reserved** Format: MBZ

HCP_TILE_STATE

HCP_TILE_STATE

Source: VideoCS

Length Bias: 2

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: 11h HCP_TILE_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: Fh — Name:
1	31:10	**Reserved** Format: MBZ
	9:5	**NumTileColumnsMinus1** Format: U5

HCP_TILE_STATE

		Specifies the number of tile columns in Ctbs per picture. Maximum of 20 columns are supported (level 6.2 restriction)
	4:0	**NumTileRowsMinus1** Format: U5 Specifies the number of tile rows in Ctbs per picture. Maximum of 22 rows are supported (level 6.2 restriction)
2..6	159:0	**Ctb column position of tile column** Format: **HCP_TILE_POSITION_IN_CTB[5]**
7..12	191:0	**Ctb row position of tile row** Format: **HCP_TILE_POSITION_IN_CTB[6]** Note that there are only 22 rows, so the most significant 16 bits of HCP_TILE_POSITION_IN_CTB[5] (31:16) are reserved
13..14	63:0	**Ctb column position MSB** Format: **HCP_TILE_POSITION_IN_CTB_MSB**
15..16	63:0	**Ctb row position MSB** Format: **HCP_TILE_POSITION_IN_CTB_MSB**

HCP_VP9_PAK_OBJECT

HCP_VP9_PAK_OBJECT

Source: VideoCS

Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = Bh
	22:16	**Media Instruction Command** Default Value: 35h HCP_VP9_PAK_OBJECT Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 0h — Name:
1	31	**IsLastSB Flag (of Tile)** Format: Enable Value 0 — Name: False Value 1 — Name: True
	30	**Reserved** Format: MBZ
	29:24	**CU_count_minus1**

		Format: U6 Number of CUs in the current SB = CU_count_minus1 + 1. Minimum, there must be 1 CU in a SB.
	23:21	**Reserved** Format: MBZ
	20:0	**split_coding_unit_flag[x0][y0]** Format: U21 Bit 20: Split_flag_level0 Bit 19:16: Split_flag_level1 [19:16] is in raster order. Bit16 is for partition0 in raster order. Bit 15:12: Split_flag_level2_level1part3 Split flags for bit19 partition. [15:12] is in raster order. Bit12 is for partition0 in raster order. Bit 11:8: Split_flag_level2_level1part2 Split flags for bit18 partition. [11:8] is in raster order. Bit8 is for partition0 in raster order. Bit 7:4: Split_flag_level2_level1part1 Split flags for bit17 partition. [7:4] is in raster order. Bit4 is for partition0 in raster order. Bit 3:0: Split_flag_level2_level1part0 Split flags for bit16 partition. [3:0] is in raster order. Bit0 is for partition0 in raster order.
2	31:16	**Current SB Y Addr)** Format: U16
	15:0	**Current SB X Addr** Format: U16

HCP_VP9_PIC_STATE

HCP_VP9_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 Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = Bh
	22:16	**Media Instruction Command** Default Value: 30h HCP_VP9_PIC_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value / Name / Programming Notes: Bh – Decoder DW Length – Only Up to DW12 should be programmed for decoder 1Eh – Encoder DW Length – All DWs should be programmed for encoder
1	31:30	**Reserved** Format: MBZ
	29:16	**Frame Height In Pixels Minus 1** Format: U14 Specifies the height of the decoded picture in units of 8 pixels, which is the minimum coding block size. The decoded picture height in units of luma samples equals (FrameHeightInMinBlocksMinus1+ 1) *8 – 1 *For Encoder Partial SB:*

Kernel, on last SB (at picture edges), splits the SB into as small as possible CUs to have picture boundaries aligned to CU boundary.
Kernel does not count/include CUs outside of the picture in PakObj/CU_record respectively.
Driver sets up a SB aligned (both in X/Y direction) surface.

Value	Name
[4096,16383]	4K_TO_16K

Programming Notes

Decoder supports 16K image and 8K video. Encoder only supports up to 8K.

Value	Description
0-6	Invalid (multiple of 8 pixels)
7-8191	8-8K Pixels (Decoder and Encoder)
8192-16383	8K-16K Pixels (Decoder only)

15:14 **Reserved**

Format:	MBZ

13:0 **Frame Width In Pixels Minus 1**

Format:	U14

Specifies the width of the decoded picture in units of minimum coding block size. The decoded picture width in units of luma samples equals (FrameWidthInMinBlocksMinus1+ 1) *8 – 1 This should be programmed to a multiple of 8 pixels minus 1.
For Encoder Partial SB:
Kernel, on last SB (at picture edges), splits the SB into as small as possible CUs to have picture boundaries aligned to CU boundary.
Kernel does not count/include CUs outside of the picture in PakObj/CU_record respectively.
Driver sets up a SB aligned (both in X/Y direction) surface.

Value	Name
[4096,16383]	4K_TO_16K

Programming Notes

Decoder supports 16K image and 8K video. Encoder only supports up to 8K.

Value	Description
0-6	Invalid (multiple of 8 pixels)
7-8191	8-8K Pixels (Decoder and Encoder)
8192-16383	8K-16K Pixels (Decoder only)

2 31 **Segment ID StreamIn Enable**

Format:	Enable

Indicates SegmentID from previous frame needs to be streamIn for Segment ID prediction

Value	Name

0	Disable
1	Enable

Programming Notes
Deocder Only: SegmentIDStreamInEnable = error_resilient_mode OR intra_only OR VP9_KEY_FRAME

30 **Segment ID StreamOut Enable**

Format:	Enable

Indicates SegmentID of current frame needs to be streamOut for next frame

Value	Name
0	Disable
1	Enable

Programming Notes
Deocder Only: SegmentIDStreamOutEnable = segementation_enabled AND segmentation_update_map

29 **Lossless Mode**

Format:	Enable

This bitSet to indicate lossless coding mode.
In encoder mode, software has to set tx_mode to 4x4only and all tu_size's in CU record as 4x4 for entire frame. Software also has to program such that final_qindex=0 and final_filter_level=0 following the Quant Scale and Filter Level Table in Segmentation State section. Hardware forces Hadamard Tx when this bit is set. When Lossless Mode is on, BRC has to be off.

Value	Name
0	Normal Mode
1	Lossless Mode

28 **Segmentation Temporal Update**

Format:	Enable

Indicates whether segID is decoding from bitstream or predicted from previous frame.
In encoder Mode it should use either from previous frame or streamIn

Value	Name
0h	Decode segID from bitstream
1h	Get segID either from bitstream or from previous frame

		Programming Notes Decoder Only: For KEY_FRAME or INTRA_ONLY frame, this bit should be set to "0". Note: Driver should override this flag to "0" in KEY_FRAME or INTRA_ONLY frame even if this bit decoded from bitstream is different. This is for hardware optimization. This override does not affect bitstream decoding other than uncompressed header.
	27	**Segmentation Update Map** Format: Enable Indicates how hardware determines segmentation ID Value / Name / Description: 0h — Intra block: segment ID is zero Inter block: get segment ID from previous frame (streamIN) 1h — Intra block: decode segment ID from bitstream. Inter block: determines from segmentation_temporal_update setting
	26	**Segmentation Enabled** Format: Enable Indicate if segementation is enabled or not Value / Name: 0h — All blocks are implied to belong to segment 0 1h — SegID determination depends on segmentation_update_map setting
	25:23	**Sharpness Level** Format: U3 Specify the sharpness level, as one of regular deblocking strength control. **Programming Notes** Set to 0 to disable the use of sharpness control
	22:17	**Filter Level** Format: U6 Specify the Filter level, as one of deblocking strength control **Programming Notes** Set to 0 to disable the use of level control
	16	**Frame Parallel Decoding Mode** Indicates if parallel decoding mode is enabled. This bit should come from Uncompressed header. Together with Error Resilient mode, they decide the value of AdaptProbabilityFlag. Value / Name: 0 — Disable 1 — Enable
	15	**Error Resilient Mode** Indicates if error resilient mode is enabled. This bit should come from Uncompressed header.When error resilient is 1, Frame Parallel Decoding Mode will be 1, and Refresh Frame

Context will be 0.When error resilient is 0, Frame Parallel Decoding Mode and Refresh Frame Context read from bit stream. Together with Frame Parallel Decoding mode, they decide the value of AdaptProbabilityFlag.

Value	Name
0	Disable
1	Enable

Programming Notes

14 **Refresh Frame Context**

Indicates if Frame Context should be refresh. This bit should come from Uncompressed header

Value	Name
0	Disable
1	Enable

Programming Notes
Decoder Only

13 **Last Frame Type**

It indicates the frame type of previous frame (Key or Non-Key Frame)

Value	Name
0	Key Frame
1	Non Key Frame

Programming Notes
Not used in Encoder Mode

12 **Selectable TX Mode**

Format:	U1

Indicates if tx_mode is selectable

Value	Name
0	Encoder does not pack tu_size into bitstream. This helps reduce bitstream size further.
1	Encoder packs tu_size into bitstream.

Programming Notes
HW always picks tu_size from CU record of pak_obj. SW responsibility to set tu_size correct.

11 **Hybrid Prediction Mode**

Format:	U1

Indicates if comp_pred_mode is hybrid

Value	Name

		0 comp_prediction_mode!=HYBRID, Encoder does not pack comp_pred_mode [interpred_comp in pak_obj] into bitstream. 1 comp_prediction_mode==HYBRID, Encoder packs comp_pred_mode into bitstream. This helps reduce bitstream size further.
	10	**Use Prev in Find MV References** Format: Enable 0: Temporal MV buffer is not available for MV prediction 1: Temporal MV buffer is available for MV prediction This is set to 0 when: • The last picture has a different size • Current picture is error-resilient mode • Current picture is intra_only, or keyframe • Last picture was intra_only or keyframe • Last picture was not a displayed picture.
	9:7	**Ref Frame Sign Bias[0..2]** Format: U3 Reference Frame sign bias (not including intra reference) Bit[7] – Sign Bias of Last Frame Bit[8] – Sign Bias of Golden Frame Bit[9] – Sign Bias of AltRef Frame
	6:4	**Mcomp Filter Type** Format: U3 Indicate Motion Compensation Filter type. If set to 4, encoder uses modes in pak_obj command.

Value	Name
0h	Eight-tap
1h	Eight-tap-Smooth
2h	Eight-tap-Sharp
3h	Bilinear
4h	Switchable

	3	**Allow Hi Precision MV** Format: Enable Indicate high precision mode for Motion Vector prediction

Value	Name
0h	Normal mode
1h	High Precision mode

	2	**IntraOnly Flag** Format: Enable Indicates intra-only for inter pics. MBZ for keyframes. **Programming Notes**

Used for Non-displayable picture; SW responsibility to make sure no Inter block in pak_obj of this frame

Value	Description
0	Inter Frame use both intra/inter-blocks
1	Inter frame use only inta-blocks

1 — Adapt Probabilities Flag

Format:	Enable

Indicates that the probabilities used to decode this frame should be adapted

Value	Name
0h	0: Do not adapt (error resilient or frame_parallel_mode are set)
1h	1: Adapt (not error resilient and not frame_ parallel_mode)

Programming Notes
Only forward adaptation is supported in encoder mode.

0 — Frame Type

Format:	U1

Specifies the VP9 frame type

Value	Name
0h	Key Frame
1h	Inter Frame

3 — 31:28 — Profile Level

Format:	U4

This indicates VP9 Profile level from bitstream

Programming Notes
Profile 0, 1, 2, and 3 are supported. Profile 0: 8 bit 420 only Profile 1: 8 bit 444 (422 is NOT supported) Profile 2: 10/12 bit 420 Profile 3: 10/12 bit 444 (422 is NOT supported)

27:24 — BitDepthMinus8

Format:	U4

This indicates the bitdepth (minus 8) of the pixels

Value	Name	Programming Notes
0	Bitdepth_8	It indicates pixel bitdepth is 8. Only profile 0 is allowed in this mode. [] It indicates pixel bitdepth is 8. Only profile 0 and 1 are allowed in this

		mode.
2	Bitdepth_10	It indicates pixel bitdepth is 10. Only profile 2 is allowed in this mode. [] It indicates pixel bitdepth is 10. Only profile 2 and 3 are allowed in this mode.
4	Bitdepth_12	It indicates pixel bitdepth is 12. Only profile 2 is allowed in this mode. [] It indicates pixel bitdepth is 12. Only profile 2 and 3 are allowed in this mode. Not valid in Gen11

Programming Notes
In profile 0 and 1, only value of 0 (8 bit pixel) is allowed. In profile 2 and 3, only value of 2 and 4 (10 or 12 bit pixel) are allowed.

23:22 **Chroma Sampling Format**

Format:	U2

This indicates the chroma sampling format of the bitstream

Value	Name	Programming Notes
0	Format_420	Chroma Format 420, supported by profile 0 and 2
2	Format_444	[] Chroma Format 444, supported by Profile 1 and 3

Programming Notes
Currently only 420 and 444 are supported (in profile 0, 1, 2 and 3). All other modes are not valid. Value 0: Format 420: Profile 0 and 2 only (supported) Value 1: Format 422: Profile 1 and 3 only: Currently NOT supported Value 2: Format 444: Profile 1 and 3 only:(supported)

21 **Reserved**

Format:	MBZ

20:10 **Reserved**

Format:	MBZ

9:8 **Log2 Tile Row**

Format:	U2

This indicates the number of tile rows (log2).

Value	Name
0h	1 Tile Row
1h	2 Tile Row

		2h: 4 Tile Row
		Programming Notes: Decoder Only as encoder must use TILE_CODING_COMMAND
	7:4	**Reserved** Format: MBZ
	3:0	**Log2 Tile Column** Format: U4 This indicates the number of tile rows (log2).

Value	Name
0h	1 Tile Column
1h	2 Tile Column
2h	4 Tile Column
3h	8 Tile Column
4h	16 Tile Column
5h	32 Tile Column
6h	64 Tile Column

Programming Notes
Decoder only as encoder must use TILE_CODING_COMMAND

4	31:16	**Horizontal Scale Factor for LAST** Format: U2.14 This indicates the scaling factor between current frame and the last reference frame Set to (LastWidth * 2^14)/CurrentWidth
	15:0	**Vertical Scale Factor for LAST** Format: U2.14 This indicates the scaling factor between current frame and the last reference frame Set to (LastHeight * 2^14)/CurrentHeight
5	31:16	**Horizontal Scale Factor for GOLDEN** Format: U2.14 This indicates the scaling factor between current frame and the golden reference frame Set to (LastWidth * 2^14)/CurrentWidth
	15:0	**Vertical Scale Factor for GOLDEN** Format: U2.14 This indicates the scaling factor between current frame and the golden reference frame Set to

		(LastHeight * 2^14)/CurrentHeight"
6	31:16	**Horizontal Scale Factor for ALTREF** Format: U2.14 This indicates the scaling factor between current frame and the altref reference frame Set to (LastWidth * 2^14)/CurrentWidth
	15:0	**Vertical Scale Factor for ALTREF** Format: U2.14 This indicates the scaling factor between current frame and the altref reference frame Set to (LastHeight * 2^14)/CurrentHeight
7	31	**Reserved** Format: MBZ
	30	**Reserved** Format: MBZ
	29:16	**Last Frame Hieght In Pixels Minus 1** Format: U14 Specifies the height of the Last picture in units of pixels. The Last picture height in units of luma samples equals (LastFrameHeightInMinBlocksMinus1+ 1) Value: 0-16383 — Description: 1-16K Pixels (encoder only goes to 8K)
	15:14	**Reserved** Format: MBZ
	13:0	**Last Frame Width In Pixels Minus 1** Format: U14 Specifies the width of the Last picture in units of pixels. The Last picture width in units of luma samples equals (LastFrameWidthtInMinBlocksMinus1+ 1) Value: 0-16383 — Description: 1-16K Pixels (encoder only goes to 8K)
8	31:30	**Reserved** Format: MBZ
	29:16	**Golden Frame Height In Pixels Minus 1** Format: U14 Specifies the height of the Last picture in units of pixels. The Golden picture height in units of luma samples equals (LastFrameHeightInMinBlocksMinus1+ 1)

HCP_VP9_PIC_STATE

		Value: 0-16383 — Description: 1-16K Pixels (encoder only goes to 8K)
	15:14	**Reserved** Format: MBZ
	13:0	**Golden Frame Width In Pixels Minus 1** Format: U14 Specifies the width of the Last picture in units of pixels. The Golden picture width in units of luma samples equals (LastFrameWidthtInMinBlocksMinus1+ 1) Value: 0-16383 — Description: 1-16K Pixels (encoder only goes to 8K)
9	31:30	**Reserved** Format: MBZ
	29:16	**Altref Frame Hieght In Pixels Minus 1** Format: U14 Specifies the height of the Last picture in units of pixels. The Altref picture height in units of luma samples equals (LastFrameHeightInMinBlocksMinus1+ 1) Value: 0-16383 — Description: 1-16K Pixels (encoder only goes to 8K)
	15:14	**Reserved** Format: MBZ
	13:0	**Altref Frame Width In Pixels Minus 1** Format: U14 Specifies the width of the Last picture in units of pixels. The Altref picture width in units of luma samples equals (LastFrameWidthtInMinBlocksMinus1+ 1) Value: 0-16383 — Description: 1-16K Pixels (encoder only goes to 8K)
10	31:16	**First Partition Size in Bytes [15:0]** Format: U16 Specifies the number of bytes taken up by the first partition size which handle the probability updates **Programming Notes** Only used by Decoder
	15:8	**Reserved**

		Format: MBZ
	7:0	**Uncompressed Header Length in Bytes [7:0]** Format: U8 Specifies the number of bytes taken up by the uncompressed frame header. **Programming Notes** this field is used by decoder only
11	31:4	**Reserved**
	3	**Reserved**
	2	**Reserved**
	1	**Motion Comp Scaling Enable Bit** This bit must be set to "1" **Value** / **Name** / **Description** 1 / Enable **[Default]** / This enables Motion Comp Scaling
	0	**Reserved**
12	31:0	**Reserved** Format: MBZ
13	31:26	**Reserved** Format: MBZ
	25	**Header Insertion Enable** Format: U1 **Value** / **Name** / **Description** 0 / / No header insertion into the output bitstream buffer, before the current slice encoded bits. 1 / / Header insertion into the output bitstream buffer is present, and is before the current slice encoded bits. Must be followed by the PAK Insertion Object Command to perform the actual insertion. For VP9: Header is always present and this bit can never be Zero. As HW does the header back-annotation at the end of frame we currently cannot disable it, if header was not written by HW. Media SDK – sends 2 headers. One header has original header and second header has 0s for BRC parameters (LF refDelta, ModeDelta and BaseQindex). Driver needs to pick first header for the first pass, and second header for subsequent passes. **Programming Notes** Encoder Only
	24	**Tail Insertion Enable**

		Format: U1

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.

Must be followed by the PAK Insertion Object Command to perform the actual insertion.
Tail has to be inserted only with the last slice of frame and for VP9 only at the end of Frame.
Restriction:

Programming Notes
Encoder Only

	23:16	**Base Q Index (Same as Luma AC)** Format: U8 Added to Delta Q index of Segment Valid Values : 0..255 **Programming Notes**: Encoder Only
	15:0	**Compressed header BIN count** Format: U16 Number of bins compressed header This field is fixed insideHW **Programming Notes**: Encoder Only
14	31:21	**Reserved** Format: MBZ
	20:16	**Luma DC Q Index Delta** Format: S4 QP delta value for Luma DC Valid Values : -15..15 **Programming Notes**: Encoder Only
	15:13	**Reserved** Format: MBZ
	12:8	**ChromaDC_QindexDelta** Format: S4 QP Delta Value For Chroma DC

		Valid Values : -15..15 **Programming Notes** Encoder Only
	7:5	**Reserved** Format: MBZ
	4:0	**ChromaAC_QindexDelta** Format: S4 QP Delta Value For Chroma AC Valid Values : -15..15 **Programming Notes** Encoder Only
15	31	**Reserved** Format: MBZ
	30:24	**LF_ref_delta3** Format: S6 Loop filter level delta3 value; valid range -63...63 With this range -63..63, in the case of Base LF Level > 31 (first pass) or (Base LF Level + BRC accumulated LF Delta) > 31 (subsequent pass), the lf_ref_delta0,1,2,3 and lf_mode_delta0,1 in compressed bitstream would be in the range -31..31. **Programming Notes** Encoder Only
	23	**Reserved** Format: MBZ
	22:16	**LF_ref_delta2** Format: S6 Loop filter level delta2 value; valid range -63...63 With this range -63..63, in the case of Base LF Level > 31 (first pass) or (Base LF Level + BRC accumulated LF Delta) > 31 (subsequent pass), the lf_ref_delta0,1,2,3 and lf_mode_delta0,1 in compressed bitstream would be in the range -31..31. **Programming Notes** Encoder Only
	15	**Reserved** Format: MBZ
	14:8	**LF_ref_delta1** Format: S6 Loop filter level delta1 value; valid range -63...63 With this range -63..63, in the case of Base LF Level > 31 (first pass) or (Base LF Level + BRC accumulated LF Delta) > 31 (subsequent pass), the lf_ref_delta0,1,2,3 and lf_mode_delta0,1 in

		compressed bitstream would be in the range -31..31. **Programming Notes** Encoder Only
	7	**Reserved** Format: MBZ
	6:0	**LF_ref_delta0** Format: S6 Loop filter level delta0 value; valid range -63...63 With this range -63..63, in the case of Base LF Level > 31 (first pass) or (Base LF Level + BRC accumulated LF Delta) > 31 (subsequent pass), the lf_ref_delta0,1,2,3 and lf_mode_delta0,1 in compressed bitstream would be in the range -31..31. **Programming Notes** Encoder Only
16	31:15	**Reserved** Format: MBZ
	14:8	**LF Mode Delta 1** Format: S6 Loop filter level mode1 value; valid range -63...63 With this range -63..63, in the case of Base LF Level > 31 (first pass) or (Base LF Level + BRC accumulated LF Delta) > 31 (subsequent pass), the lf_ref_delta0,1,2,3 and lf_mode_delta0,1 in compressed bitstream would be in the range -31..31. **Programming Notes** Encoder Only
	7	**Reserved** Format: MBZ
	6:0	**LF Mode Delta 0** Format: S6 Loop filter level mode1 value; valid range -63...63 With this range -63..63, in the case of Base LF Level > 31 (first pass) or (Base LF Level + BRC accumulated LF Delta) > 31 (subsequent pass), the lf_ref_delta0,1,2,3 and lf_mode_delta0,1 in compressed bitstream would be in the range -31..31. **Programming Notes** Encoder Only
17	31:16	**BitOffsetForLFModeDelta** Format: U16 Offset from starting position of output bitstream in bits where LFModeDelta should be inserted. In BRC mode, always insert LFModeDelta. {This implies uncompressed header should have: mode_ref_delta_enabled=1 and mode_ref_delta_update=1} and The uncompressed header starting from this offset **BitOffsetForLFModeDelta** has to have the

following 16 bits format:
{Start here: 1'b1, mode_delta_0[6:0], 1'b1, mode_delta_1[6:0]} and
BitOffsetForLFModeDelta = BitOffsetForLFRefDelta + 32.
Encoder Only

15:0 BitOffsetForLFRefDelta

Format:	U16

Offset from starting position of output bitstream in bits where LFRefDelta should be inserted.
In BRC mode, always insert LFRefDelta. {This implies uncompressed header should have: mode_ref_delta_enabled=1 and mode_ref_delta_update=1} and
The uncompressed header starting from this offset **BitOffsetForLFRefDelta** has to have the following 32 bits format:
{Start here: 1'b1, ref_delta_0[6:0], 1'b1, ref_delta_1[6:0], 1'b1, ref_delta_2[6:0], 1'b1, ref_delta_3[6:0]}.
Encoder Only

18 31:16 BitOffsetForLFLevel

Format:	U16

Offset from starting position of output bitstream in bits where LFLevel should be inserted.
Encoder Only

15:0 BitOffsetForQindex

Format:	U16

Offset from starting position of output bitstream in bits where Qindex should be inserted.
Encoder Only

19 31:27 Reserved

Format:	MBZ

26 FrameSzUnderStatusEn - FrameBitRateMinReportMask

Format:	U1

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

Value	Name	Description
0	Disable	Do not update bit 2 (Frame Bit Count Violate -- under run) of HCP_VP9_IMAGE_STATUS control register.
1	Enable	Set bit 2 (Frame Bit Count Violate -- under run) of HCP_VP9_IMAGE_STATUS control register if the total frame level bit counter is less than or equal to Frame Bit Rate Minimum limit.

		Programming Notes Encoder Only
	25	**FrameSzOverStatusEn - FrameBitRateMaxReportMask** Format: U1 This is a mask bit controlling if the condition of frame level bit count exceeds FrameBitRateMax. **Value / Name / Description** 0 / Disable / Do not update bit 1 of HCP_VP9_IMAGE_STATUS control register. 1 / Enable / Set bit 1 of HCP_VP9_IMAGE_STATUS control register if the total frame level bit counter is greater than or equal to Frame Bit Rate Maximum limit. **Programming Notes** Encoder Only
	24:18	**Reserved** Format: MBZ
	17	**Reserved** Format: MBZ
	16	**NonFirstPassFlag** This signals the current pass is not the first pass. It will imply designate HW behavior. **Value / Name / Description** 0 / Disable / If it is initial-Pass, this bit is set to 0. 1 / Enable / For subsequent passes, this bit is set to 1. **Programming Notes** Encoder Only
	15:0	**Reserved** Format: MBZ
20	31	**FrameBitrateMaxUnit** Format: U1 This field is the Frame Bitrate Maximum Limit Units. **Value / Name / Description**

0	Byte	32byte unit
1	KiloByte	4Kbyte unit

Programming Notes
Encoder Only

30:14 **Reserved**

Format:	MBZ

13:0 **FrameBitRateMax**

Format:	U14

This field is the Frame Bitrate Maximum Limit. This field along with FrameBitrateMaxUnit determines maximum allowed bits in a frame before multi-pass gets triggered (when enabled). In other words, multi-pass is triggered when the actual frame byte count exceeds this value.

0-512KB The programmable range is 0-512KB when FrameBitrateMaxUnit is 0.

0-64MB The programmable range is 0-64Mbyte when FrameBitrateMaxUnit is 1.

Programming Notes
Encoder Only

21 31 **FrameBitrateMinUnit**

Format:	U1

This field is the Frame Bitrate Maximum Limit Units.

Value	Name	Description
0	Byte	32byte unit
1	KiloByte	4Kbyte unit

Programming Notes
Encoder Only

30:14 **Reserved**

Format:	MBZ

13:0 **FrameBitRateMin**

Format:	U14

		This field is the Frame Bitrate Minimum Limit. This field along with FrameBitrateMinUnit 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. 0-512KB The programmable range is 0-512KB when FrameBitrateMinUnit is 0. 0-64MB The programmable range is 0-64Mbyte when FrameBitrateMinUnit is 1. **Programming Notes** Encoder Only
22..23	63:0	**FrameDeltaQindexMax** Format: FrameDeltaQindexLFRange Frame level delta Qindex which should be used in case FrameSize - FrameBitRateMax in the range of ((FrameDeltaQindexLFMaxRange[n] * FrameBitRateMax»5)), FrameDeltaQindexLFMaxRange[n+1] * FrameBitRateMax»5)). Each DelatQindexMax value is 8-bit with S7 format **Programming Notes** If n == 7, DeltaQpMaxRange is infinity. Encoder Only
24	31:0	**FrameDeltaQindexMin** Format: FrameDeltaQindexLFRange Frame level delta Qindex which should be used in case FrameSize - FrameBitRateMin in the range of ((FrameDeltaQindexLFMinRange[n] * FrameBitRateMin»5)), FrameDeltaQindexLFMinRange[n+1] * FrameBitRateMin»5)). Each DelatQindexMin value is 8-bit with S7 format **Programming Notes** If n == 3, FrameDeltaQindexLFMaxRange is zero. (n>3 is not supported) Encoder Only
25..26	63:0	**FrameDeltaLFMax** Format: FrameDeltaQindexLFRange Frame level delta Loop Filter Level which should be used in case FrameSize - FrameBitRateMax in the range of ((FrameDeltaQindexLFMaxRange[n] * FrameBitRateMax»5)), FrameDeltaQindexLFMaxRange[n+1] * FrameBitRateMax»5)). Each delta_lf_max is 7 bits with S6 format **[bits 7, 15, 23, 31,.....63 are reserved]** **Programming Notes** If n == 7, FrameDeltaQindexLFMaxRange is infinity. Encoder Only

27	31:0	**FrameDeltaLFMin** Format: FrameDeltaQindexLFRange Frame level delta Loop Filter Level which should be used in case FrameSize - FrameBitRateMin in the range of ((FrameDeltaQindexLFMinRange[n] * FrameBitRateMin»5)), FrameDeltaQindexLFMinRange[n+1] * FrameBitRateMin»5)). Each delta_lf_min is 7 bits with S6 format **[bits 7, 15, 23, 31 are reserved]** **Programming Notes** If n == 3, FrameDeltaQindexLFMaxRange is zero. (n>3 is not supported) Encoder Only
28..29	63:0	**FrameDeltaQindexLFMaxRange** Format: FrameDeltaQindexLFRange Condition: FrameDeltaQindexLFMaxRange[n] >= FrameDeltaQindexLFMaxRange[n-1] This field is to calculate ranges for Frame level delta Qindex, specifically Frame level delta Qindex[n] and Frame level delta Qindex[n+1]. **Programming Notes** If n == 0, FrameDeltaQindexLFMaxRange is zero. Encoder Only
30	31:0	**FrameDeltaQindexLFMinRange** Format: FrameDeltaQindexLFRange Condition: FrameDeltaQindexLFMinRange[n] >= FrameDeltaQindexLFMinRange[n-1] This field is to calculate ranges for Frame level delta Qindex, specifically Frame level delta Qindex[n] and Frame level delta Qindex[n+1]. **Programming Notes** If n == 0, FrameDeltaQindexLFMinRange is zero. Encoder Only
31	31:30	**MinFrameSizeUnits** Format: U2 This field is the Minimum Frame Size Units

Value	Name	Description
0	4Kb	Minimum Frame Size is in 4Kbytes.
1	16Kb	Minimum Frame Size is in 4Kbytes.
2	Comaptibility Mode	
3	6 Bytes	

HCP_VP9_PIC_STATE

		Programming Notes Encoder Only
	29:16	**Reserved** Format: MBZ
	15:0	**MinFramSize** Format: U16 Minimum Frame Size [15:0] (in Word, 16-bit)(Encoder Only) Minimum Frame Size is specified to compensate for intel Rate Control Currently zero fill (no need to perform emulation byte insertion) is done at the last slice of a picture. It is needed for CBR. Intel encoder parameter, not part of DXVA. The caller should always make sure that the value, represented by Minimum Frame Size, is always less than maximum frame size FrameBitRateMax. This field is reserved in Decode mode. **Programming Notes** Programmable range is 0..(2^16-1) * 2^12 when MinFrameSizeUnits is 0. (4KB unit) Programmable range is 0..(2^16-1) * 2^14 when MinFrameSizeUnits is 1. (16KB unit) Encoder Only
32	31:16	**Reserved** Format: MBZ
	15:0	**BitOffsetForFirstPartitionSize** Format: U16 Offset from starting position of output bitstream in bits where First Partition Size should be inserted. **Programming Notes** Encoder Only

HCP_VP9_SEGMENT_STATE

HCP_VP9_SEGMENT_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 Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = Bh
	22:16	**Media Instruction Command** Default Value: 32h HCP_VP9_SEGMENT_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 6h — Name:
1	31:3	**Reserved** Format: MBZ
	2:0	**Segment ID** Format: U3 The segment ID of the DWORDS following this one
2	31:4	**Reserved** Format: MBZ
	3	**Segment Reference Enabled**

HCP_VP9_SEGMENT_STATE

		Format: Enable For encoder: When "Segment Reference Enabled" is set to 1, Software (Kernel) needs to program all PUs of this segment accordingly. This means: CU record PU refframe0="Segment Reference" bit[2:1], refframe1=0, and interpred_comp=single.
	2:1	**Segment Reference** Format: U2 Indicates reference frame for this segment. **Programming Notes** If the current frame is a KEY frame or INTRA_ONLY frame, this field should be set to INTRA for all segments.
	0	**Segment Skipped** Format: Enable Indicates skip mode for this segment. **Programming Notes** If set to 1, all delta coefficients and MVs within CU of this segment should be forced to zero in HW. No block less than 8x8 size allowed segmentSkipped If set to 0, skipcoeff_flag should be coded as normal
3	31:30	**Reserved** Format: MBZ
	29:24	**FilterLevelRef1Mode1** Exists If: //Decoder mode only Format: U6 Indicates final filter level for Ref1 (Last Frame) and Mode 1.
	23:22	**Reserved** Format: MBZ
	21:16	**FilterLevelRef1Mode0** Exists If: //Decoder mode only Format: U6 Indicates final filter level for Ref1 (Last Frame) and Mode 0.
	15:14	**Reserved** Format: MBZ
	13:8	**FilterLevelRef0Mode1** Exists If: //Decoder mode only Format: U6 Indicates final filter level for Ref0 (Last Frame) and Mode 1.

	7:6	**Reserved** Format: MBZ
	5:0	**FilterLevelRef0Mode0** Exists If: //Decoder mode only Format: U6 Indicates final filter level for Ref0 (Last Frame) and Mode 0.
4	31:30	**Reserved** Format: MBZ
	29:24	**FilterLevelRef3Mode1** Exists If: //Decoder mode only Format: U6 Indicates final filter level for Ref3 (Last Frame) and Mode 1.
	23:22	**Reserved** Format: MBZ
	21:16	**FilterLevelRef3Mode0** Exists If: //Decoder mode only Format: U6 Indicates final filter level for Ref3 (Last Frame) and Mode 0.
	15:14	**Reserved** Format: MBZ
	13:8	**FilterLevelRef2Mode1** Exists If: //Decoder mode only Format: U6 Indicates final filter level for Ref2 (Last Frame) and Mode 1.
	7:6	**Reserved** Format: MBZ
	5:0	**FilterLevelRef2Mode0** Exists If: //Decoder mode only Format: U6 Indicates final filter level for Ref2 (Last Frame) and Mode 0.
5	31	**Reserved**

HCP_VP9_SEGMENT_STATE

		Format: MBZ
	30:16	**Luma AC Quant Scale (Decode mode Only)** Format: U15 Indicates final value of Luma AC Quantized Scale value. Value: [4,29247] — Name: Valid_Range
	15	**Reserved** Format: MBZ
	14:0	**Luma DC Quant Scale (Decode mode Only)** Format: U15 Indicates final value of Luma DC Quantized Scale value. Value: [4,21387] — Name: Valid_Range
6	31	**Reserved** Format: MBZ
	30:16	**Chroma AC Quant Scale (Decode mode Only)** Format: U15 Indicates final value of Chroma AC Quantized Scale value. Value: [4,29247] — Name: Valid_Range
	15	**Reserved** Format: MBZ
	14:0	**Chroma DC Quant Scale (Decode mode Only)** Format: U15 Indicates final value of Chroma DC Quantized Scale value. Value: [4,21387] — Name: Valid_Range
7	31:23	**Reserved** Format: MBZ
	22:16	**Segment LF Level Delta (Encode mode Only)** Format: S6 Indicates the Loop Filter Delta for this segment. Must be 0 when segmentation_enabled == 0. Range -63..63

HCP_VP9_SEGMENT_STATE

	15:9	**Reserved** Format: MBZ
	8:0	**Segment QIndex Delta (encode mode only)** Format: S8 Indicates the QIndex delta for this segment. Must be 0 when segmentation_enabled == 0. Range -255..255

HCP_WEIGHTOFFSET_STATE

HCP_WEIGHTOFFSET_STATE

Source: VideoCS

Length Bias: 2

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

This slice level command is issued in both the encoding and decoding processes, if the weighted_pred_flag or weighted_bipred_flag equals one. If zero, then this command is not issued.
Weight Prediction Values are provided in this command. Only Explicit Weight Prediction is supported in encoder.
For P-Slice, this command is issued only once together with HCP_REF_IDX_STATE Command for L0 list. For B-Slice, this command can be issued up to two times together with HCP_REF_IDX_STATE Command, one for L0 list and one for L1 list.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: 13h HCP_WEIGHTOFFSET_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 28h — Name: 40
1	31:1	**Reserved**

HCP_WEIGHTOFFSET_STATE

		Format: MBZ
	0	**RefPicListNum** Format: U1 Value / Name: 0 – Reference Picture List 0 1 – Reference Picture List 1
2..17	511:0	**LumaOffsets** Format: HCP_WEIGHTOFFSET_LUMA_ENTRY[16]
18..33	511:0	**ChromaOffsets** Format: HCP_WEIGHTOFFSET_CHROMA_ENTRY[16]
34..41	255:0	**ChromaOffsetsExt** Format: HCP_WEIGHTOFFSET_CHROMA_EXT_ENTRY[8]

HEVC_VP9_RDOQ_STATE

HEVC_VP9_RDOQ_STATE

Source: VideoCS

Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h Format: Opcode PARALLEL_VIDEO_PIPE
	28:27	**Pipeline** Default Value: 2h Format: Opcode MFX_COMMON
	26:23	**Opcode** Default Value: 7h Format: Opcode Codec/Engine Name = HCP = 7h
	22:21	**SubOpA** Default Value: 0h Format: Opcode
	20:16	**SubOpB** Default Value: 8h Format: Opcode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Total Length - 2 Value: 3Ch — Name: DWORD_COUNT_n **[Default]** — Description: Excludes DWord (0,1)
1	31	**Disable HTQ performance fix0** set to disable performance optimizations done by doubling LNZ and OSR storage Set to 1, to go back to single LNZ and OSR (no optimization) Set to 0, to use double buffer LNZ and OSR

HEVC_VP9_RDOQ_STATE		
	30	**Disable HTQ performance fix1** set to disableperformance optimization done to save 1clk while switching from EBB to GTRAM storage. This is critical for IntraLoop performance Set to 1, disable optimization Set to 0, enable optimization
	29:0	**Reserved** Format: MBZ
2..33	1023:0	**IntraLumaLambda** Format: **HEVC_VP9_RDOQ_LAMBDA_FIELDS[32]**
34..65	1023:0	**IntraChromaLambda** Format: **HEVC_VP9_RDOQ_LAMBDA_FIELDS[32]**
66..97	1023:0	**InterLumaLambda** Format: **HEVC_VP9_RDOQ_LAMBDA_FIELDS[32]**
98..129	1023:0	**InterChromaLambda** Format: **HEVC_VP9_RDOQ_LAMBDA_FIELDS[32]**

HI8DS Render Target Write MSD

MSD_RTW_HI8DS - HI8DS Render Target Write MSD

Source: EuSubFunctionRenderDataPort
Length Bias: 1
Family: Other
Group: Render Target R/W

DWord	Bit	Description
0	31	**Reserved** Format: MBZ Ignored
	30	**Message Precision Subtype** Default Value: 0h Format: Opcode Full precision data message
	29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present**

MSD_RTW_HI8DS - HI8DS Render Target Write MSD

	Format:	MDC_MHP

If set, indicates that the message includes the 2-register header.

18 Per-Coarse Pixel PS outputs enable

Format:	Enable

This bit indicates the render target write is a coarse pixel write.

Programming Notes

This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.

17:14 Message Type

Default Value:	0Ch
Format:	Opcode

Render Target Write message

13 Per-Sample PS Enable

Format:	Enable

If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot.

Programming Notes

This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE.
When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0.
When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples).
When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.

12 Last Render Target Select

Format:	Enable

MSD_RTW_HI8DS - HI8DS Render Target Write MSD

		This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.
	11	**Slot Group Select** Format: **MDC_RT_SGS** This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
	10:8	**Render Target Message Subtype** Default Value: 3h Format: Opcode SIMD8 dual source message. Use slots [15:8] for pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [31:24] are referenced instead of [15:8].
	7:0	**Binding Table Index** Format: **MDC_BTS** Specifies the Binding Table Index for the message

If

if - If

Source: Eulsa

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 32-bit exit code <JIP> and <UIP>. If <branch_ctrl> is set, then the JIP points to the first join instruction within the if block. If <branch_ctrl> is not set, <JIP> should point to the instruction right after the matching else instruction if it exists, otherwise <JIP> should point to the endif instruction. <UIP> should always point to the endif instruction. When a jump occurs, this value is added to IP pre-increment. In GEN instruction binary, <JIP> and <UIP> are at location <src0> & <src1> and must be of type D (signed dword integer).

Format:

```
[(pred)] if (exec_size JIP UIP <branch_ctrl>
```

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) imm32 imm32 <branch_ctrl>
```

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	N	N	N

DWord	Bit	Description
0..3	127:96	**JIP**

<table>
<tr><th colspan="4">if - If</th></tr>
<tr><td></td><td></td><td colspan="2">
<table>
<tr><td>Format:</td><td>S31</td></tr>
</table>
The byte-aligned jump distance if a jump is taken for the channel.
</td></tr>
<tr><td></td><td>95:64</td><td colspan="2">UIP
<table>
<tr><td></td><td></td></tr>
<tr><td>Format:</td><td>S31</td></tr>
</table>
The byte aligned jump distance if a jump is taken for the instruction.
</td></tr>
<tr><td></td><td>63:32</td><td colspan="2">Operand Control
<table>
<tr><td>Format:</td><td>EU_INSTRUCTION_OPERAND_CONTROLS</td></tr>
</table>
</td></tr>
<tr><td></td><td>31:0</td><td colspan="2">Header
<table>
<tr><td>Format:</td><td>EU_INSTRUCTION_HEADER</td></tr>
</table>
</td></tr>
</table>

Illegal

illegal - Illegal

Source: Eulsa

Length Bias: 4

The Illegal Opcode Exception Enable flag in cr0.1 is normally set so the normal processing of an illegal opcode is to transfer control to the System Routine. Instruction dispatch treats any unused 8-bit opcode (including bit 7 of the instruction, reserved for future opcode expansion) as if it is the illegal opcode. The illegal opcode is zero because that byte value is more likely than most to be read via a wayward instruction pointer. The illegal instruction is an instruction only in the same way that a NULL pointer in software is a pointer. Both are special values indicating invalid instances.

Format:

```
illegal
```

Restriction

The illegal instruction takes no instruction options.

Syntax

illegal

Pseudocode

{ Set the Illegal Opcode Exception Status bit in cr0.1. if (Illegal Opcode Exception Enable is set in cr0.1) { Transfer control to the System Routine (return address to AIP, IP = SIP). } }

Predication	Conditional Modifier	Saturation	Source Modifier
N	N	N	N

DWord	Bit	Description
0..3	127:7	**Reserved** Format: MBZ
	6:0	**Opcode** Format: EU_OPCODE

Integer Subtraction with Borrow

subb - Integer Subtraction with Borrow

Source: Eulsa

Length Bias: 4

The subb instruction performs component-wise subtraction of src0 and src1 and stores the results in dst, it also stores the borrow into acc. If the operation produces a borrow (src0 < src1), write 0x00000001 to acc, else write 0x00000000 to acc.

Format:

```
[(pred)] subb[.cmod] (exec_size) dst src0 src1
```

Restriction

AccWrEn is required. The accumulator is an implicit destination and thus cannot be an explicit destination operand.

Syntax

```
[(pred)] subb[.cmod] (exec_size) reg reg reg
[(pred)] subb[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
          for ( n = 0; n < exec_size; n++ ) {
             if ( WrEn.chan[n] ) {
                dst.chan[n] = src0.chan[n] - src1.chan[n];
                acc.chan[n] = borrow(src.chan[n] - src1.chan[n]);
             }
          }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	N

Src Types	Dst Types
UD	UD

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

subb - Integer Subtraction with Borrow			
		Format:	EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header**	
		Format:	EU_INSTRUCTION_HEADER

Join

join - Join

Source: Eulsa

Length Bias: 4

The join instruction makes the inactive channels active at the join IP if those channels are predicated. Any deactivated channels due to a goto instruction match the join IP are activated (qualified with predicates at join). If no IP is matched at this join, the program goes to the next IP with the active channels which followed the program path up to the join instruction. If no active channels are present after executing the join instruction, the program jumps to the offset specified by JIP instead of next IP. The join instruction is used in conjunction with a goto instruction. The join activates channels that are deactivated by the goto instruction. See the goto instruction for the deactivation rules. The goto and join instructions enable unstructured program control flow. These instructions must be used with additional care where dangling channels can result without proper compiler checks, meaning that it is expected that programs will navigate through these paths to reactivate the channels. Hardware does not provide native checks or reconvergence. The following table describes the 32-bit JIP. In GEN binary, JIP is at location src1 and must be of type D (signed DWord integer). JIP must be an immediate operand and is a signed 32-bit number. This value is added to IP pre-increment. If SPF is ON, none of the PcIP are updated.

Format:

```
[(pred)] join (exec_size) JIP
```

Programming Notes

An index of 0 is an infinite loop.

Restriction

The {NoMask} instruction option must be specified.

The index data type must be D (Signed DWord Integer).

Syntax

```
[(pred)] join (exec_size) imm32
```

Pseudocode

```
Evaluate(WrEn);
          for ( n = 0; n < exec_size; n++ ) {
             if (WrEn.chan[n] ) {  // for the predicated channels and the remaining
channels
                PcIP[n] = IP + 1;
             }
          }

          if ( PcIP != (IP + 1) ) {  // for all channels when no channels are activated
and no other active channels
             Jump(IP + JIP);
```

join - Join

```
}
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

DWord	Bit	Description
0..3	127:96	**JIP** Format: S31 Jump Target Offset. The relative offset in bytes if a jump is taken for the instruction.
	95	**Source 0 Address Immediate [9] Sign Bit**
	94:91	**Src1.SrcType** Format: **SrcType**
	90:89	**Src1.RegFile** Format: **RegFile**
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align16') Format: **EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN16**
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1') Format: **EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN1**
	63:32	**Operand Control** Format: **EU_INSTRUCTION_OPERAND_CONTROLS**
	31:0	**Header** Format: **EU_INSTRUCTION_HEADER**

Jump Indexed

jmpi - Jump Indexed

Source: Eulsa

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. 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.
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.

Format:

```
[(pred)] jmpi (1) index {NoMask}
```

Programming Notes

An index of 0 does nothing, continuing execution with the next instruction.

An index of -16 (if the jmpi instruction is in native format) or -8 (if the jmpi instruction is in compact format) is an infinite loop on the jmpi instruction.

Restriction

The execution size must be 1.

MaskCtrl must be specified.

The index data type must be D (Signed DWord Integer).

QtrCtrl must not be used for jmpi instruction.

Syntax

```
[(pred)] jmpi (1) reg32 {NoMask}
[(pred)] jmpi (1) imm32 {NoMask}
```

Pseudocode

```
Evaluate(WrEn); if ( WrEn != 0 ) { Jump(IP + 1 + index ); // IP + 1 is a pseudocode idiom
for the IP of the following instruction. }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

jmpi - Jump Indexed

Src Types
D

DWord	Bit	Description
0..3	127:96	**JIP** Format: S31 Jump Target Offset. The relative offset in bytes if a jump is taken for the instruction.
	95	**Source 0 Address Immediate [9] Sign Bit**
	94:91	**Src1.SrcType** Format: SrcType
	90:89	**Src1.RegFile** Format: RegFile
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align16') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN16
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1') Format: EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN1
	63:32	**Operand Control** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

Leading Zero Detection

lzd - Leading Zero Detection

Source: Eulsa

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
```

Syntax

```
[(pred)] lzd[.cmod] (exec_size) reg reg
[(pred)] lzd[.cmod] (exec_size) reg imm32
```

Pseudocode

```
Evaluate(WrEn);
   for ( n = 0; n < exec_size; n++ ) {
      if ( WrEn.chan[n] ) {
         UD udScalar = src0.chan[n];
         UD cnt = 0;
         while ( (udScalar & (1 « 31)) == 0 && cnt != 32 ) {
            cnt ++;
            udScalar = udScalar « 1;
         }
         dst.chan[n] = cnt;
      }
   }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types
D, UD	UD

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

lzd - Leading Zero Detection		
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

LO8DS Render Target Write MSD

MSD_RTW_LO8DS - LO8DS Render Target Write MSD

Source: EuSubFunctionRenderDataPort
Length Bias: 1
Family: Other
Group: Render Target R/W

DWord	Bit	Description
0	31	**Reserved** Format: MBZ Ignored
	30	**Message Precision Subtype** Default Value: 0h Format: Opcode Full precision data message
	29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present**

MSD_RTW_LO8DS - LO8DS Render Target Write MSD

	Format:	MDC_MHP
	If set, indicates that the message includes the 2-register header.	
18	**Per-Coarse Pixel PS outputs enable**	
	Format:	Enable
	This bit indicates the render target write is a coarse pixel write.	
	Programming Notes	
	This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.	
17:14	**Message Type**	
	Default Value:	0Ch
	Format:	Opcode
	Render Target Write message	
13	**Per-Sample PS Enable**	
	Format:	Enable
	If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot.	
	Programming Notes	
	This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE. When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0. When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples). When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case. This bit MUST be 0 for Dual Source Messages.	
12	**Last Render Target Select**	
	Format:	Enable
	This bit must be set on the last render target write message sent for each group of pixels. For	

MSD_RTW_LO8DS - LO8DS Render Target Write MSD

		single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.
	11	**Slot Group Select** Format: MDC_RT_SGS This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
	10:8	**Render Target Message Subtype** Default Value: 2h Format: Opcode SIMD8 dual source message. Use slots [7:0] for pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [23:16] are referenced instead of [7:0].
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Logic And

and - Logic And

Source: Eulsa

Length Bias: 4

The and instruction performs component-wise logic AND operation between src0 and src1 and stores the results in dst. Register source operands can use source modifiers. Any source modifier is numeric, optionally changing a source value s to -s, abs(s), or -abs(s) before the AND operation. Any source modifier is logical, optionally changing a source value s to ~s (inverting all source bits). This capability allows expressions like a AND (NOT b) to be calculated with one instruction. This operation does not produce sign or overflow conditions. Only the .e/.z or .ne/.nz conditional modifiers should be used.

Format:

```
Source modifier is not allowed if source is an accumulator.
```

Restriction

Source modifier is not allowed if source is an accumulator.

Syntax

```
[(pred)] and[.cmod] (exec_size) reg reg reg
         [(pred)] and[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
          for ( n = 0; n < exec_size; n++ ) {
            if ( WrEn.chan[n] ) {
              dst.chan[n] = src0.chan[n] & src1.chan[n];
            }
          }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	N	Y

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

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

and - Logic And		
		Format: EU_INSTRUCTION_SOURCES_REG_IMM
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

Logic Not

not - Logic Not

Source: Eulsa

Length Bias: 4

Description

The not instruction performs logical NOT operation (or one's complement) of src0 and storing the results in dst. This operation does not produce sign or overflow conditions. Only the .e/.z or .ne/.nz conditional modifiers should be used.

A register source operand can use a source modifier: Any source modifier is logical, optionally changing a source value s to ~s (inverting all source bits). Such a source modifier is not particularly useful with the not instruction, as it changes the effect of not to just copying bits.

Format:

```
[(pred)] not[.cmod] (exec_size) dst src0
```

Restriction

Source modifier is not allowed if source is an accumulator.

Syntax

```
[(pred)] not[.cmod] (exec_size) reg reg
[(pred)] not[.cmod] (exec_size) reg imm32
```

Pseudocode

```
Evaluate(WrEn);
for ( n = 0; n < exec_size; n++ ) {
   if ( WrEn.chan[n] ) {
      dst.chan[n] = ~ src0.chan[n];
   }
}
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	N	Y

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

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG

not - Logic Not		
	127:64	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: **EU_INSTRUCTION_SOURCES_IMM32**
	63:32	**Operand Controls** Format: **EU_INSTRUCTION_OPERAND_CONTROLS**
	31:0	**Header** Format: **EU_INSTRUCTION_HEADER**

Logic Or

or - Logic Or

Source: Eulsa

Length Bias: 4

Description

The or instruction performs component-wise logic OR operation between src0 and src1 and stores the results in dst. This operation does not produce sign or overflow conditions. Only the .e/.z or .ne/.nz conditional modifiers should be used.

Register source operands can use source modifiers: Any source modifier is logical, optionally changing a source value s to ~s (inverting all source bits). This capability allows expressions like a OR (NOT b) to be calculated with one instruction.

Format:

```
[(pred)] or[.cmod] (exec_size) dst src0 src1
```

Restriction

Source modifier is not allowed if source is an accumulator.

Syntax

```
[(pred)] or[.cmod] (exec_size) reg reg reg
[(pred)] or[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
for ( n = 0; n < exec_size; n++ ) {
   if ( WrEn.chan[n] ) {
      dst.chan[n] = src0.chan[n] | src1.chan[n];
   }
}
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	N	Y

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

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG

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

Logic Xor

xor - Logic Xor

Source: Eulsa

Length Bias: 4

Description

The xor instruction performs component-wise logic XOR operation between src0 and src1 and stores the results in dst. This operation does not produce sign or overflow conditions. Only the .e/.z or .ne/.nz conditional modifiers should be used.

Register source operands can use source modifiers: Any source modifier is logical, optionally changing a source value s to ~s (inverting all source bits). This capability allows expressions like a XOR (NOT b) to be calculated with one instruction.

Format:

```
[(pred)] xor[.cmod] (exec_size) dst src0 src1
```

Restriction

Source modifier is not allowed if source is an accumulator.

Syntax

```
[(pred)] xor[.cmod] (exec_size) reg reg reg
[(pred)] xor[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
for ( n = 0; n < exec_size; n++ ) {
   if ( WrEn.chan[n] ) {
      dst.chan[n] = src0.chan[n] ^ src1.chan[n];
   }
}
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	N	Y

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

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG

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

MEDIA_CURBE_LOAD

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 | 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	**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 64-byte aligned.
3	31:0	**CURBE Data Start Address** Format: DynamicStateOffset[31:0] **Description** Specifies the 64-byte aligned address of the CURBE data. This pointer is relative to the **Dynamics Base Address**. **Value**: [0, FFFFFFFFh] **Name**:

MEDIA_INTERFACE_DESCRIPTOR_LOAD

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

MEDIA_INTERFACE_DESCRIPTOR_LOAD

		[32,2048] | [1,64] interface descriptor entries
3	31:0	**Interface Descriptor Data Start Address** Format: | DynamicStateOffset[31:0]INTERFACE_DESCRIPTOR_DATA **Description** This bit specifies the 64-byte aligned address of the Interface Descriptor data. This pointer is relative to the Dynamics Base Address. **Value** | **Name** [0, FFFFFFFFh] |

MEDIA_OBJECT_GRPID

MEDIA_OBJECT_GRPID

Source: RenderCS

Length Bias: 2

The MEDIA_OBJECT_GRPID command is a variation of MEDIA_OBJECT which includes a group id which is used to allocate and track Barriers and Shared Local Memory. The Interface Descriptor is used to specify how much SLM is needed and how many threads will be reporting to the Barrier. All MEDIA_OBJECT_GRPIDs with the same group id should have the same interface descriptor and be dispatched to the same Tslice – the dispatcher will ensure this if Force Destination = 0, but software must ensure this if Force Destination = 1. Software should also ensure that all the threads needed for the Barrier will fit into a Tslice, or the Barrier will never be satisfied. Either SLM or a barrier must be used with MEDIA_OBJECT_GRPID, if neither is needed then a MEDIA_OBJECT must be used instead.
MEDIA_OBJECT_GRPID supports the GPGPU version of payload delivery – either indirect or CURBE can be split between the threads in a group (per-thread payload), as well as a section which is sent to all threads (cross-thread payload). See the GPGPU payload section. For indirect, the same pointer must be sent with all the commands associated with the thread group for payload splitting to work properly. Inline data is not split, but the payload attached to each command is sent with that thread. Only one of inline, indirect, or CURBE is allowed, but at least one form of payload must be sent.
MEDIA_STATE_FLUSH with the watermark bit must be placed between groups created by MEDIA_OBJECT_GRPID. The Interface Descriptor associated with the watermark must match the Interface Descriptor used for the following group.

Programming Notes

The ICL_LP 1x8x8 configuration only has VME units on 4 of the 8 subslices.
Before using this command to dispatch kernels that use VME assets, theR_PWR_CLK_STATE register (0x20C8) must be set to use 1 slice ([18]=1 and [17:12]=01h) to ensure that the dispatches are limited to Slice0 which has the 4 subslices that have VME units. See R_PWR_CLK_STATE for more details.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Media Command Pipeline** Default Value: 2h Media Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h MEDIA_OBJECT_GRPID Format: OpCode
	23:16	**Media Command Sub-Opcode** Default Value: 6h MEDIA_OBJECT_GRPID SubOp

MEDIA_OBJECT_GRPID

		Format: OpCode
	15:0	**DWord Length** Format: =n Total Length - 2 Excludes DWords 0,1 **Generic Mode:** DWord Length = N+5, where N is in the range of [0,504]. The maximum is 504 DW (equivalent to 63 8-DW registers). 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. **Value** / **Name** 5h / DWORD_COUNT_n **[Default]** [5,112] / Min_Max
1	31:8	**Reserved**
	7:6	**Reserved** Format: MBZ
	5:0	**Interface Descriptor Offset** Format: U6 This field specifies the offset from the interface descriptor base pointer to the interface descriptor which will be applied to this object. It is specified in units of interface descriptors. **Value** / **Name** [0,30] /
2	31:25	**Reserved** Format: MBZ
	24	**Reserved**
	23	**End of Thread Group** This bit indicates that this dispatch is the last for the current thread group. **Restriction** By default, when one SS in a DSS is disabled (e.g. for die recovery configurations), the maximum thread group size is limited to 40 threads/group. When all SS in all DSS are enabled, the maximum thread group size remains 112 threads/group.
	22	**Reserved**
	21	**Reserved** Format: MBZ
	20:19	**Reserved**

MEDIA_OBJECT_GRPID

		Format: MBZ
	18:17	**Reserved** Format: MBZ
	16:0	**Indirect Data Length** Format: U17 In bytes This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. This field must have the same alignment as the Indirect Object Data Start Address. It must be DQWord (32-byte) aligned. As the indirect data are sent directly to URB, range is limited to(URB Entry Allocation Size)*(Number of URB Entries) in the MEDIA_VFE_STATE command.
3	31:0	**Indirect Data Start Address** Format: GraphicsAddress[31:0] **Description** This field specifies the Graphics Memory starting address of the data to be loaded into the kernel for processing. This pointer is relative to the **Indirect Object Base Address**. Hardware ignores this field if indirect data is not present. Alignment of this address depends on the mode of operation. It is the 64-byte aligned address of the indirect data. **Value** / **Name** / **Description** [0-512MB] / / Bits 31:29 MBZ
4	31:25	**Reserved** Format: MBZ
	24:16	**Y Position** Format: U9 This field provides the Y position of the block to be dispatched. It is sent the kernel via the R0 header, but is not otherwise used by hardware.
	15:9	**Reserved** Format: MBZ
	8:0	**X Position** Format: U9 This field provides the X position of the block to be dispatched. It is sent to the kernel via the R0 header, but is not otherwise used by hardware.

MEDIA_OBJECT_GRPID

5	31:24	**RESERVED** Format: MBZ
	23:16	**Block Color** Format: U8 This field specifies the color value associated with the block to be dispatched. It is sent to the kernel via the R0 header, but is not otherwise used by hardware.
	15:8	**Reserved** Format: MBZ
	7:0	**Reserved** Format: MBZ
6	31:0	**GroupID** A unique identifying number which describes the threads which share a barrier and/or SLM. Reuse of numbers is allowed as long as the old group is not currently running.
7..n	31:0	**Inline Data** The format of this data is specified by software. Hardware does not interpret this data; it merely passes it to the kernel for processing. The total size for the inline data and indirect data must not exceed 112 registers.

MEDIA_OBJECT

MEDIA_OBJECT

Source: RenderCS

Length Bias: 2

Programming Notes

The ICL_LP 1x8x8 configuration only has VME units on 4 of the 8 subslices.
Before using this command to dispatch kernels that use VME assets, the R_PWR_CLK_STATE register (0x20C8) must be set to use 1 slice ([18]=1 and [17:12]=01h) to ensure that the dispatches are limited to Slice0 which has the 4 subslices that have VME units. See R_PWR_CLK_STATE for more details.

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	**Reserved** Format: MBZ
	14:0	**DWord Length** Default Value: [4,116] Min_Max 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**

MEDIA_OBJECT

	7:6	**Reserved** Format: MBZ
	5:0	**Interface Descriptor Offset** Format: U6 This field specifies the offset from the interface descriptor base pointer to the interface descriptor which will be applied to this object. It is specified in units of interface descriptors.
2	31	**Reserved** Format: MBZ
	30:27	**Reserved** Format: MBZ
	26:25	**Slice Destination Select MSBs** These bits are the MSBs of the slice destination select field. The LSB bits are 20:19. Only slices that are available can be selected.
	24	**Thread Synchronization** This field when set indicates that the dispatch of the thread originated from this command is based on the "spawn root thread" message. Value: 0 — No thread synchronization Value: 1 — Thread dispatch is synchronized by the 'spawn root thread' message
	23	**Reserved** Format: MBZ
	22	**Force Destination** If set, bits 20:17 are used to determine the destination of this dispatch, if clear the destination will be chosen based on load.
	21	**Reserved** Format: MBZ
	20:17	**Subslice Destination Select** Format: U4 This field selects the SubSlice that this thread must be sent to. This configuration has only 1 slice.

Ignored if **Force Destination** = 0.

Value	Name	Description
[0-7]	Subslice ID	Valid subslice number in this configuration.
[8-15]	Reserved	Illegal values. Results undefined if used.

	16:0	**Indirect Data Length** Format: U17 In bytes This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. This field must have the same alignment as the Indirect Object Data Start Address. It must be DQWord (32-byte) aligned. As the indirect data are sent directly to URB, range is limited to (URB Entry Allocation Size)*(Number of URB Entries) in the MEDIA_VFE_STATE command. When both inline and indirect data are fetched for this command, enough room should be allowed for at least 1 copy of the indirect and 1 copy of the inline data in the URB.
3	31:0	**Indirect Data Start Address** Format: GraphicsAddress[31:0]

Description
This field specifies the Graphics Memory starting address of the data to be loaded into the kernel for processing. This pointer is relative to the **Indirect Object Base Address**. Hardware ignores this field if indirect data is not present. Alignment of this address depends on the mode of operation.
This field specifies the 64-byte aligned address of the indirect data.

Value	Name
[0,512MB]	

Programming Notes
Bits 31:29 MBZ

4	31:25	**Reserved** Format: MBZ
	24:16	**Y Position** Format: U9 This field provides the Y position of the block dispatched by this command. It is sent to the kernel via the R0 header but is not otherwise used by hardware.
	15:9	**Reserved** Format: MBZ

MEDIA_OBJECT

	8:0	**X Position** Format: U9 This field provides the X position of the block dispatched by this command. It is sent to the kernel via the R0 header but is not otherwise used by hardware.
5	31:24	**Reserved** Format: MBZ
	23:16	**Block Color** Format: U8 This field specifies the color for the block being dispatched. It is sent to the kernel via the R0 header, but is not otherwise used by hardware.
	15:8	**Reserved** Format: MBZ
	7:0	**Reserved** Format: MBZ
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_WALKER

MEDIA_OBJECT_WALKER

Source: RenderCS

Length Bias: 2

Programming Notes

The ICL_LP 1x8x8 configuration only has VME units on 4 of the 8 subslices.
Before using this command to dispatch kernels that use VME assets, theR_PWR_CLK_STATE register (0x20C8) must be set to use 1 slice ([18]=1 and [17:12]=01h) to ensure that the dispatches are limited to Slice0 which has the 4 subslices that have VME units. See R_PWR_CLK_STATE for more details.

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	**Reserved** Format: MBZ
	14:0	**DWord Length** Default Value: [15,112] Min_Max 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**

MEDIA_OBJECT_WALKER

	7:6	**Reserved** Format: Reserved
	5:0	**Interface Descriptor Offset** Format: U6 This field specifies the offset from the interface descriptor base pointer to the interface descriptor which will be applied to this object. It is specified in units of interface descriptors.
2	31:25	**Reserved** Format: MBZ
	24	**Thread Synchronization** This field when set indicates that the dispatch of the thread originated from this command is based on the "spawn root thread" message. Value / Name 0 – No thread synchronization 1 – Thread dispatch is synchronized by the 'spawn root thread' message
	23:22	**Masked Dispatch** Format: U2 Enable the masking of the dispatch of individual threads based on a bitmask read from CURBE, and specifies the pitch of the CURBE surface. If enabled, CURBE will not be used for thread payload. Value / Name / Description 00b – Masked Dispatch Disabled 01b – Masked Dispatch with 128-bit pitch in CURBE 10b – Masked Dispatch with 256-bit pitch in CURBE 11b – Masked Dispatch with 512-bit pitch in CURBE
	21	**Reserved** Format: MBZ
	20:17	**Reserved** Format: MBZ
	16:0	**Indirect Data Length** Format: U17 in bytes Description 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 a URB entry, the range is

		limited to the URB Entry Allocation Size specified in the MEDIA_VFE_STATE command.
3	31:0	**Indirect Data Start Address** This field specifies the Graphics Memory starting address of the data to be loaded into the kernel for processing. This pointer is relative to the **Indirect Object Base Address.** Hardware ignores this field if indirect data is not present. Alignment of this address depends on the mode of operation. It is the 64-byte aligned address of the indirect data

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

4	31:0	**Reserved**

Format:	MBZ

5	31:8	**Group ID Loop Select**

This bit field chooses which of the nested loops of the walker are used to identify threads which share a group id and therefore a shared barrier and SLM. The programmer must ensure that each group will fit into a single subslice. When barriers are enabled every group must have the same number of threads matching the number specified in the Interface Descriptor.

Value	Name	Description
0	No_Groups	Groups are not created, barriers and SLM are not allocated
2	InnerLocal_Groups	Each complete iteration of the Inner local loop and Color loop defines a group, the group id is the concatenation of the Outer global loop to the Mid local loop execution counts.
3	MidLocal_Groups	Each complete iteration of the Mid local loop and lower loops defines a group, the group id is the concatenation of the Outer global loop to the Outer local loop execution counts.
4	OuterLocal_Groups	Each complete iteration of the Outer local loop and lower loops defines a group, the group id is the concatenation of the Outer global loop and the Inner global loop execution counts.
5	InnerGlobal_Groups	Each complete iteration of the Inner global loop and lower loops defines a group, the group id is the Outer global loop execution count.
Others	Reserved	

Restriction
When media thread groups are formed (i.e. Group ID Loop Select is non-zero), software must ensure the size of the group does not exceed the maximum number of threads in a DSS (112).
By default, when one SS in a DSS is disabled (e.g. for die recovery configurations), the maximum thread group size is limited to 40 threads/group. When all SS in all DSS are enabled, the maximum thread group size remains 112 threads/group.

MEDIA_OBJECT_WALKER

	7:0	**Reserved** Format: MBZ
6	31:24	**Color Count Minus One** Format: U8 This field specifies the number of repeat of the inner most loop of the walker. Each repeated walk position is assigned with an incremental Color number. The Color number together with the X and Y position of the thread is used for dependency scoreboard control. Usage Example: This allows multiple sets of dependency threads to be dispatched.
	23:21	**Reserved** Format: MBZ
	20:16	**Middle Loop Extra Steps** Format: U5
	15:14	**Reserved** Format: MBZ
	13:12	**Local Mid-Loop Unit Y** Format: S1
	11:10	**Reserved** Format: MBZ
	9:8	**Mid-Loop Unit X** Format: S1
	7:0	**Reserved** Format: MBZ
7	31:28	**Reserved** Format: MBZ
	27:16	**Global Loop Exec Count** Format: U12
	15:12	**Reserved** Format: MBZ
	11:0	**Local Loop Exec Count** Format: U12
8	31:27	**Reserved** Format: MBZ
	26:16	**Block Resolution Y**

		Format: U11 Vertical resolution of the local loop.
	15:11	**Reserved** Format: MBZ
	10:0	**Block Resolution X** Format: U11 Horizontal resolution of the local loop.
9	31:27	**Reserved** Format: MBZ
	26:16	**Local Start Y** Format: U11 Starting vertical position of the local loop.
	15:11	**Reserved** Format: MBZ
	10:0	**Local Start X** Format: U11 Starting horizontal position of the local loop.
10	31:0	**Reserved** Format: MBZ
11	31:28	**Reserved** Format: MBZ
	27:16	**Local Outer Loop Stride Y** Format: S11 Vertical stride of the local outer loop, in 2's complement.
	15:12	**Reserved** Format: MBZ
	11:0	**Local Outer Loop Stride X** Format: S11 Horizontal stride of the local outer loop, in 2's complement.
12	31:28	**Reserved** Format: MBZ

MEDIA_OBJECT_WALKER		
	27:16	**Local Inner Loop Unit Y** Format: S11 Vertical stride of the local inner loop, in 2's complement.
	15:12	**Reserved** Format: MBZ
	11:0	**Local Inner Loop Unit X** Format: S11 Horizontal stride of the local inner loop, in 2's complement.
13	31:27	**Reserved** Format: MBZ
	26:16	**Global Resolution Y** Format: U11 Vertical resolution of the global loop.
	15:11	**Reserved** Format: MBZ
	10:0	**Global Resolution X** Format: U11 Horizontal resolution of the global loop.
14	31:28	**Reserved** Format: MBZ
	27:16	**Global Start Y** Format: S11 Starting vertical location of the global loop, in 2's complement.
	15:12	**Reserved** Format: MBZ
	11:0	**Global Start X** Format: S11 Starting horizontal location of the global loop, in 2's complement.
15	31:28	**Reserved** Format: MBZ
	27:16	**Global Outer Loop Stride Y** Format: S11

MEDIA_OBJECT_WALKER		
		Vertical stride of the global outer loop, in 2's complement.
	15:12	**Reserved** Format: MBZ
	11:0	**Global Outer Loop Stride X** Format: S11 Horizontal stride of the global outer loop, in 2's complement.
16	31:28	**Reserved** Format: MBZ
	27:16	**Global Inner Loop Unit Y** Format: S11 Vertical stride of the global inner loop, in 2's complement.
	15:12	**Reserved** Format: MBZ
	11:0	**Global Inner Loop Unit X** Format: S11 Horizontal stride of the global inner loop, in 2's complement.
17..n	31:0	**Inline Data**

MEDIA_STATE_FLUSH

MEDIA_STATE_FLUSH

Source: RenderCS

Length Bias: 2

This command updates the Message Gateway state. In particular, it updates the state for a selected Interface Descriptor.
This command can be considered same as a MI_Flush except that only media parser will get flushed instead of the entire 3D/media render pipeline. The command should be programmed prior to new Media state, curbe and/or interface descriptor commands when switching to a new context or programming new state for the same context. With this command, pipelined state change is allowed for the media pipe.
Be cautious when using this command when child_present flag in the media state is enabled. This is because that CURBE state as well as Interface Descriptor state are shared between root threads and child threads. Changing these states while child threads are generated on the fly may cause unexpected behavior. Combining with MI_ARB_ON/OFF command, it is possible to support interruptability with the following command sequence where interrupt may be allowed only when MI_ARB_ON_OFF is ON:

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:24	**Media Command Opcode** Default Value: 0h MEDIA_STATE_FLUSH Format: OpCode
	23:16	**SubOpcode** Default Value: 4h MEDIA_STATE_FLUSH SubOp Format: OpCode
	15:0	**DWord Length** Format: =n Total Length - 2 Value / Name / Description 0h / DWORD_COUNT_n **[Default]** / Excludes DWord (0,1)

MEDIA_STATE_FLUSH

1	31:9	**Reserved** Format: MBZ
	8	**Reserved**
	7	**Flush to GO** Format: Enable This bit indicates that the write data out of this thread group should be flushed to the point where it is visible to following commands.
	6	**Reserved** Format: MBZ
	5:0	**Interface Descriptor Offset** Format: U6 This field specifies the offset from the interface descriptor base pointer to the interface descriptor which describes what resources are required to meet the watermark.

MEDIA_VFE_STATE

MEDIA_VFE_STATE

Source: RenderCS

Length Bias: 2

A stalling PIPE_CONTROL is required before MEDIA_VFE_STATE unless the only bits that are changed are scoreboard related: Scoreboard Enable, Scoreboard Type, Scoreboard Mask, Scoreboard * Delta. For these scoreboard related states, a MEDIA_STATE_FLUSH is sufficient.

- MEDIA_STATE_FLUSH (optional, only if barrier dependency is needed)
- MEDIA_INTERFACE_DESCRIPTOR_LOAD (optional)

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:24	**Media Command Opcode** Default Value: 0h MEDIA_VFE_STATE Format: OpCode
	23:16	**SubOpcode** Default Value: 0h MEDIA_VFE_STATE SubOp Format: OpCode
	15:0	**DWord Length** Format: =n Total Length - 2 Value / Name / Description 07h / 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.**
	9:8	**Reserved** Format: MBZ

MEDIA_VFE_STATE

	7:4	**Stack Size** Value: [0,11] — indicating [1KBytes, 2MBytes] **Programming Notes** Since the stack uses the upper portion of the scratch space, **Stack Size =< Per Thread Scratch Space**
	3:0	**Per Thread Scratch Space** Format: U4 Specifies the amount of scratch space allowed to be used by each thread. Value: [0,11] — indicating [1k bytes, 2 Mbytes]: 0=1k, 1=2k, 2=4k, 3=8k ... 11=2M **Programming Notes** The driver must allocate enough contiguous scratch space, pointed to by the Scratch Space Pointer, to ensure that all threads have their own per-thread scratch space. Scratch Space Size = (MEDIA_VFE_STATE::Per Thread Scratch Space) * (MEDIA_VFE_STATE::Maximum Number Of Threads). If a fused configuration has fewer threads than the native POR configuration, the scratch space allocation is based on the number of threads in the base native POR configuration.
2	31:16	**Reserved** Format: MBZ
	15:0	**Scratch Space Base Pointer High** Format: GeneralStateOffset[47:32] This field specifies the high 16 bits of starting address of the Scratch Space Base Pointer
3	31:16	**Maximum Number of Threads** Format: U16-1 **Description** Range: [0, 2^16-1], representing [1, 2^16] threads. Normally set to the maximum number of threads: (# EUs) * (# threads/EU). See *Graphics Processing Engine* for listing of #EUs and #threads in each device. See Programming Restrictions here for additional limitations. Restriction : Starting with this configuration, the Maximum Number of Threads must be set to (#EU * 8) for

GPGPU dispatches.
Although there are only 7 threads per EU in the configuration, the FFTID is calculated as if there are 8 threads per EU, which in turn requires a larger amount of Scratch Space to be allocated by the driver.

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
[1,128]		[1,128] Entries

Programming Notes
Please note that 0 is not allowed for this field.

7 **Reserved**

Format:	MBZ

6 **Disable Slice 0 Subslice 2**

Format:	Disable

When set, GPGPU and Media thread dispatches will not be generated to subslice 2 in slice 0. When clear, subslice 2 will receive GPGPU and Media thread dispatches.

Programming Notes
MEDIA_STATE_FLUSH command should be precede the changing of this control.

Restriction
When this control is set, use thread-group or command-level preemption (not mid-thread preemption).

5:3 **Reserved**

Format:	MBZ

2 **Dispatch Load Balance**

This bit determines how media threads are dispatched between the various dual subslices. GPGPU threads are not impacted by this bit.

Value	Name	Description
1	Color LSB	When this value is used the threads are split into two groups depending on the LSB of the color value for media threads. One group will be sent to even

MEDIA_VFE_STATE

				dual subslices and the other to odd dual subslices. The least loaded active dual subslice available will selected for threads in that group. This allows color to be used to separate workloads with different operations to get better cache coherency. If media with groups is being used then each group is kept in the same dual subslices.
		0	Least Loaded	When this value is used the threads are sent to the least loaded dual subslice of all active dual subslices. If media with groups is being used then each group is kept in the same dual subslice.
	1:0	**Reserved**		
4	31:8	**Reserved**		
	7:0	**Maximum Number of Dual-Subslices**		

This field determines the maximum number of dual subslices which are allowed to be used. Lowering this value below the maximum number of dual subslices available will reduce the memory footprint for context and scratch space, as well as potentially saving power. A zero in this field indicates that all available dual subslices should be enabled. Setting this field to a value larger or equal to the available dual subslices will also enable all of them.

5	31:16	**URB Entry Allocation Size**

Format:	U16

Specifies the length of each URB entry used by the unit, in 256-bit (32 byte) register increments. Each URB entry is rounded up to be 64-byte aligned. ROB address for URB starts after CURBE Allocated region.

Programming Notes
When Inline data is used with MEDIA_OBJECT or MEDIA_OBJECT_WALKER, then the URB entry allocation size must match the Inline data size. If Indirect data is being used with MEDIA_OBJECT or GPGPU_WALKER then the allocation size must be sufficient for the Indirect data. If both Inline and Indirect are being used, then the allocation size must match the sum of the Inline and Indirect.
The total size of URB used by VFE must be less than number of bytes allocated for the URB in the L3 banks (value in L3CNTLREG / 32 bytes per entry). The total size of URB used by VFE = Number of Interface Descriptors (=64) + **CURBE Allocation Size** + (**Number of URB Entrie**s * (roundup(**URB Entry Allocation Size**, 64)))

Restriction
The total URB allocation size cannot be greater than 1024kB.

	15:0	**CURBE Allocation Size**

Format:	U16

Specifies the total length allocated for CURBE, in 256-bit (32 byte) register increments. Each URB entry is rounded up to be 64-byte aligned. ROB address for CURBE starts at address 64.

MEDIA_VFE_STATE

		Programming Notes CURBE Allocation Size should be 0 for GPGPU and Media workloads that use indirect instead of CURBE. See programming notes and restrictions on the **URB Entry Allocation Size** for limitations on the total URB size that apply to CURBE Allocation Size.
6	31:30	**Reserved** Format: MBZ
	29:16	**Reserved** Format: MBZ
	15:8	**Reserved** Format: MBZ
	7:0	**Reserved** Format: MBZ
7	31:0	**Reserved** Format: MBZ
8	31:0	**Reserved** Format: MBZ

Media Block Read MSD

MSD1R_MB - Media Block Read MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Other
Group: Media Block R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 04h Format: Opcode Media Block Read message
	13:11	**Reserved**

MSD1R_MB - Media Block Read MSD

		Format: MBZ Ignored
	10:8	**Vertical Line Stride Override** Format: MDC_VLSO If enabled, specifies the Vertical Line Stride and Vertical Line Stride Offset override fields.
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Media Block Write MSD

MSD1W_MB - Media Block Write MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Other
Group: Media Block R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 0Ah Format: Opcode Media Block Write message
	13:11	**Reserved**

MSD1W_MB - Media Block Write MSD

		Format: MBZ Ignored
	10:8	**Vertical Line Stride Override** Format: MDC_VLSO If enabled, specifies the Vertical Line Stride and Vertical Line Stride Offset override fields.
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Media Transpose Read MSD

MSD1R_TT - Media Transpose Read MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Other
Group: Transpose Read

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 00h Format: Opcode Transpose Read message
	13:8	**Reserved**

MSD1R_TT - Media Transpose Read MSD

		Format: MBZ Ignored
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Memory Fence MSD

MSD_MEMFENCE - Memory Fence MSD

Source: EuSubFunctionDataPort0
Length Bias: 1
Family: Other
Group: Memory Fence

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP Indicates that the message requires a header.
	18	**Legacy Message** Default Value: 0h Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 07h Format: Opcode Memory Fence message

MSD_MEMFENCE - Memory Fence MSD

	13	**Commit** Format: Enable Specifies whether control is returned to the thread only after the fence has been honored. **Value / Name / Description** 1 / Enabled **[Default]** / The commit writeback register is always required to guarantee ordering. 0 / Reserved / The commit writeback register is always required to guarantee ordering.
	12:9	**L3 Flush** The L3 Flush control is one of the following GSYNC signals. **Value / Name / Description** 0h / Disabled **[Default]** / The L3 caches are not flushed. 08h / RW Data / Causes the L3 to flush any RW data. 04h / Constant Data / Causes the L3 to invalidate any Constant data. 02h / Texture Data / Causes the L3 to invalidate any Texture data. 01h / Instructions / Causes the L3 to invalidate all GPU instruction caches. **Programming Notes** If multiple caches need to be flushed, the commands need to be sent separately. When the memory fence completes, the GSYNC has been started, but may not yet be completed. To know when the GSYNC is completed, Issue any read to the L3 cache after an L3 Flush operation and wait for that data to be returned.
	8	**L1 Flush Data** Format: Enable When set, invalidate this subslice's L1 read-only data cache.
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies whether global memory or shared local memory (SLM) is fenced with this operation. **Value / Name / Description** 0FEh / SLM / Only SLM is fenced with this operation. Global memory is not fenced. Restriction : The L3 Flush and L1 Flush fields are ignored when SLM memory is selected. 00h / Global / Only global memory is fenced with this operation. SLM memory is not

MSD_MEMFENCE - Memory Fence MSD				
			Memory **[Default]**	fenced. Performance : When a program needs to guarantee that all global memory writes are globally observable before a thread retires, a Memory Fence operation is used immediately before the EOT message.

MFC_AVC_PAK_OBJECT

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**

MFC_AVC_PAK_OBJECT

		000Ah: 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 data fetching is disabled - subsequently, the Indirect PAK-MV Data Start Address field is ignored. This field must have the same alignment as the Indirect PAK-MV Data Start Address. This field must be DW aligned (since each MV is 4 bytes in size).Driver has to derived this field from MVsize (MVquantity in DXVA, exact size) *4 bytes per MV.
2	31:29	**Reserved** Format: MBZ
	28:0	**Indirect PAK-MV Data Start Address Offset** This field specifies the memory starting address (offset) of the MV data to be fetched into PAK Subsystem for processing. This pointer is relative to the MFC Indirect PAK-MV Object Base Address.Hardware ignores this field if indirect data is not present, i.e. the Indirect PAK-MV Data Length is set to 0. It is a Dword aligned address in all AVC encoding configuration, since each MV is 4 bytes in size. **Value**: [0,512MB) — **Name**:
3..10	255:0	**Inline Data** Format: U32[8] 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.
11	31:0	**Reserved** Format: MBZ
12..23	383:0	**VDEnc Mode Inline Data** Format: U32[12] In VDEnc mode, PAK gets inline MVs. These DWs are placed in the PAK Object command in-order to facilitate PAK stand-alone validation mode. Its definition is described in the next section.

MFC_JPEG_HUFF_TABLE_STATE

MFC_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. Two Huffman tables for luma and chroma will be sent to H/W, and chroma table is used for both U and V.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFC_JPEG_HUFF_TABLE_STATE Format: OpCode
	26:24	**Media Command Opcode** Default Value: 7h JPEG Format: OpCode
	23:21	**SubOpcode A** Default Value: 2h Common Format: OpCode
	20:16	**SubOpcode B** Default Value: 3h MEDIA_ Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 0AEh Excludes DWord (0,1) Format: =n Total Length - 2
1	31:1	**Reserved** Format: MBZ
	0	**Huff Table ID** Format: U1 Huffman table destination identifier will specify one of two destinations at the encoder into which the Huffman table must be stored.

MFC_JPEG_HUFF_TABLE_STATE

Value	Name	Description
0		Huffman table 0
1		Huffman table 1

2..13	383:0	**DC_TABLE** Format: 3Bytes: Byte0 for Code length, Byte1 and Byte2 for Code word, and Byte3 for dummy 12 categories with code length and code word. Each run/size has 1-byte code length, and 2-byte code word.
14..175	5183:0	**AC_TABLE** Format: 3Bytes: Byte0 for Code length, Byte1 and Byte2 for Code word, and Byte3 for dummy 162 run/size with code length and code word. Each run/size has 1-byte code length, and 2-byte code word.

MFC_JPEG_SCAN_OBJECT

MFC_JPEG_SCAN_OBJECT

Source: VideoCS

Length Bias: 2

Encoder Only

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFC_JPEG_SCAN_OBJECT Format: OpCode
	26:24	**Media Command Opcode** Default Value: 7h JPEG_ENC Format: OpCode
	23:21	**SubOpcode A** Default Value: 2h Format: OpCode
	20:16	**SubOpcode B** Default Value: 9h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 001h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:26	**Reserved** Format: MBZ
	25:0	**MCU Count** Format: U26 This field indicates the number of MCUs in the Scan. MCU Count = M_x *X* M_y The number of MCUs in a row: M_x= *(X + (H_1 *8 -1)) / (H_1 *8)* The number of MCUs in a column: M_y= *(Y + (V_1*8 -1)) / (V_1*8)X*: The number of samples per line in Y-image *Y*: The number of lines in Y-image *H1*: Horizontal sampling factor of Y-image in the Frame header *V1*: Vertical sampling factor of Y-image in the Frame header

MFC_JPEG_SCAN_OBJECT

2	31:25	**Reserved** Format: MBZ
	24:22	**Huffman AC Table** AC Huffman table destination selector specifies one of two possible AC table destinations for each Y, U, V, or R, G, B. The AC Huffman tables must have been loaded in destination 0 and 1 by the time of issuing MFC_JPEG_HUFF_TABLE_STATE Command. If AC table 0 is used for Y and AC table 1 is used for U and V, it will be set to 110b. If AC table 0 is used for R, G, and B, it will be set to 000b and so on. Refer to the table below for the summary of actions.

Value	Name	Description
0XXb	Bit24 (V0)	The third image component must use the AC table 0.
1XXb	Bit24 (V1)	The third image component must use the AC table 1.
X0Xb	Bit23 (U0)	The second image component must use the AC table 0.
X1Xb	Bit23 (U1)	The second image component must use the AC table 1.
XX0b	Bit22 (Y0)	The first image component must use the AC table 0.
XX1b	Bit22 (Y1)	The first image component must use the AC table 1.

Restriction
When InputSurfaceFormatYUV = RGB, because the order of input image components can be RGB, GBR, BGR,\ or YUV, **Bit22** is used for the first image component, **Bit23** is used for the second image component, and **Bit24** is used for the third image component.

	21	**Reserved** Format: MBZ
	20:18	**Huffman DC Table** DC Huffman table destination selector specifies one of two possible DC table destinations for each Y, U, V, or R, G, B. The DC Huffman tables shall have been loaded in destination 0 and 1 by the time of issuing MFC_JPEG_HUFF_TABLE_STATE Command. if DC table 0 is used for Y and DC table 1 is used for U and V, it will be set to 110b. If DC table 0 is used for R, G, and B, it will be set to 000b and so on. Refer to the table below for the summary of actions.

Value	Name	Description
0XXb	Bit20 (V0)	The third image component must use the DC table 0.
1XXb	Bit20 (V1)	The third image component must use the DC table 1.
X0Xb	Bit19 (U0)	The second image component must use the DC table 0.
X1Xb	Bit19 (U1)	The second image component must use the DC table 1.

XX0b	Bit18 (Y0)	The first image component must use the DC table 0.
XX1b	Bit18 (Y1)	The first image component must use the DC table 1.

Restriction
When InputSurfaceFormatYUV = RGB, because the order of input image components can be RGB, GBR, BGR, YUV, **Bit18** is used for the first image component, **Bit19** is used for the second image component, and **Bit20** is used for the third image component.

17 **Head Present Flag**

If this flag is set to 0, then no MFC_JPEG_PAK_INSERT_OBJECT commands will be sent. If this flag is set to 1, then one or more MFC_JPEG_PAK_INSERT_OBJECT commands will be sent after MFC_JPEG_SCAN_OBJECT command.

Value	Name	Description
0		No insertion into the output bitstream buffer before Scan encoded bitstream
1		Headers, tables, App data insertion into the output bitstream buffer. HW will insert the insertion data before the Scan encoded bitstream.

16 **Is Last Scan**

If this flag is set, then HW will insert EOI (0xFFD9) to the end of Scan encoded bitstream.

Value	Name	Description
0		Not the last Scan.
1		Indicates that the current Scan is the last one.

15:0 **Restart Interval**

Format:	U16

Specifies the number of MCUs in an ECS, except for the last ECS. Restart maker is inserted periodically and it separates the two neighboring ECSs.

Value	Name
0-FFFFh	

Programming Notes
A value of '0' implies that the Scan Data has a single ECS.

MFC_MPEG2_PAK_OBJECT

MFC_MPEG2_PAK_OBJECT

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)

MFC_MPEG2_PAK_OBJECT		
		Format: =n Total Length - 2
1..8	255:0	**Inline Data** Format: U32[8] All the required MB level controls and parameters for encoding are captured as inline data of the MFC_MPEG2_PAK_OBJECT command. It has a fixed size of 8 DWs. Its definition is described in the next section

MFC_MPEG2_SLICEGROUP_STATE

MFC_MPEG2_SLICEGROUP_STATE

Source: VideoCS

Length Bias: 2

This is a slice group level command and can be issued multiple times within a picture that is comprised of multiple slice groups. The same command is used for AVC encoder (PAK mode) and decoder (VLD and IT modes).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_MPEG2_SLICEGROUP_STATE Format: OpCode
	26:24	**Media Command Opcode** Default Value: 3h MPEG2 Format: OpCode
	23:21	**SubOpcode A** Default Value: 2h MEDIA_ Format: OpCode
	20:16	**SubOpcode B** Default Value: 3h MEDIA_ Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 6h Excludes DWord (0,1) Format: =n Total Length - 2
1	31	**MbRateCtrlFlag- RateControlCounterEnable (Encoder-only)** 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. Note: To reset MB level rate control (QRC), we need to set both bits MbRateCtrlFlag and MbRateCtrlReset to 1 in the new slice

MFC_MPEG2_SLICEGROUP_STATE

		Value / **Name** 0h — Disable 1h — Enable
	30	**MbRateCtrlReset- ResetRateControlCounter (Encoder-only)** To reset the bit allocation accumulation counter to 0 to restart the rate control. **Value** / **Name** / **Description** 0h — Disable — Not reset 1h — Enable — reset
	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**

MFC_MPEG2_SLICEGROUP_STATE

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	**BitstreamOutputFlag - Compressed BitStream Output Disable Flag (Encoder-only)**

Value	Name	Description
0h	Enable	enable the writing of the output compressed bitstream
1h	Disable	disable the writing of the output compressed bitstream

17 **HeaderPresentFlag - Header Insertion Present in Bitstream (Encoder-only)**

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

16 **SliceData PresentFlag - SliceData Insertion Present in Bitstream (Encoder-only)**

Value	Name	Description
0h	Disable	no Slice Data insertion into the output bitstream buffer
1h	Enable	Slice Data insertion into the output bitstream buffer is present.

15 **TailPresentFlag - Tail Insertion Present in bitstream (Encoder-only)**

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

14	**FirstSliceHdrDisabled** when this is on, the first slice header of the slice group is expected to be provided by the user via insertion command. PAK HW will skip it.
13	**IntraSlice** intra slice value included in slice headers, when IntraSliceFlag = 1.
12	**IntraSliceFlag** intra slice flag included in slice headers

MFC_MPEG2_SLICEGROUP_STATE

	11:8	**Reserved** Format: MBZ for SliceID extension
	7:4	**SliceID[3:0] (Encoder-only)** To identify the output data (coding information record) returned for rate control from PAK to ENC and VPP
	3:2	**Reserved** Format: MBZ for StreamID extension
	1:0	**StreamID[1:0] (Encoder-only)** To identify the output data (coding information record) returned for rate control from PAK to ENC and VPP
2	31:24	**NextSgMbYcnt - also NextStartVertPos** Vertical count of the first MB in the next slice group (Encoder-only)Note: This field restricts total number of MB in the Y direction to 255 or less.
	23:16	**NextSgMbXcnt - also NextStartHorzPos** BitFieldDesc
	15:8	**FirstMbYcnt - also CurrStartVertPos** Format: U8 also CurrStartVertPos, Vertical count of the first MB in the current slice group (Encoder-only)
	7:0	**FirstMbXcnt - also CurrStartHorzPos** Format: U8 Horizontal count of the first MB in the current slice group (Encoder-only)
3	31:9	**Reserved** Format: MBZ
	8	**SliceGroupSkip** Exists If: //Encoder Only Format: U1 All macroblocks are skipped
	7:6	**Reserved** Format: MBZ
	5:0	**SliceGroupQp** Exists If: //Encoder Only

MFC_MPEG2_SLICEGROUP_STATE

		Format: U6 Initial slice quality parameter
4	31:29	**Reserved** Format: MBZ
	28:0	**BitstreamOffset - Indirect PAK-BSE Data Start Address (Write)** Exists If: //Encoder Only This field specifies the memory starting address (offset) to write out the compressed bitstream data from the BSE processing. This pointer is relative to the MFC Indirect PAK-BSE Object Base Address. It is a byte-aligned address for the AVC bitstream data in both CABAC/CAVLC Modes. For Write, there is no need to have a data length field. It is assumed the global memory bound check specified in the IND_OBJ_BASE_ADDRESS command (Indirect PAK-BSE Object Access Upper Bound) will take care of any illegal write access. This field is only valid for AVC encode mode. Value: [0,512MB) — Name:
5	31:24	**MaxQpNegModifier - Magnitude of QP Max Negative Modifier (Encoder-only)** Format: U8 This field specifies the lower limit of the QP modifier. Value: [0, 51] — Name:
	23:16	**MaxQpPosModifier - Magnitude of QP Max Positive Modifier (Encoder-only)** Format: U8 This field specifies the upper limit of the QP modifier. Value: [0, 51] — Name:
	15:12	**ShrinkParam - Shrink Resistance (Encoder-only)** Format: U4 This field specifies the additional points added each time decreased correction is invoked. Value: [0, 15] — Name:
	11:8	**Shrinkaram - Shrink Init (Encoder-only)** Format: U4 This field specifies the initial points required to trip decreased control. Value: [0, 15] — Name:
	7:4	**GrowParam - Grow Resistance (Encoder-only)** Format: U4

MFC_MPEG2_SLICEGROUP_STATE

		This field specifies the additional points added each time increased correction is invoked. **Value** | **Name** [0, 15] |
	3:0	**GrowParam - Grow Init (Encoder-only)** Format: | U4 This field specifies the initial points required to trip increased control. **Value** | **Name** [0, 15] |
6	31:24	**Reserved** Format: | MBZ
	23:20	**CorrectPoints - Correct 6 (Encoder-only)** Format: | U4 This field specifies the points used in the lowermost RC region when sum_act <= sum_min. **Value** | **Name** [0, 15] |
	19:16	**CorrectPoints - Correct 5 (Encoder-only)** Format: | U4 This field specifies the points used in the fifth RC region when sum_act > sum_min but <= lower_midpt. **Value** | **Name** [0, 15] |
	15:12	**CorrectPoints - Correct 4 (Encoder-only)** Format: | U4 This field specifies the points used in the fourth RC region when sum_act > lower_midpt but <= sum_target. **Value** | **Name** [0, 15] |
	11:8	**CorrectPoints - Correct 3 (Encoder-only)** Format: | U4 This field specifies the points used in the third RC region when sum_act > sum_target but <= upper_midpt. **Value** | **Name** [0, 15] |
	7:4	**CorrectPoints - Correct 2 (Encoder-only)** Format: | U4 This field specifies the points used in the second RC region when sum_act > upper_midpt but <= sum_max.

MFC_MPEG2_SLICEGROUP_STATE

		Value / **Name** [0, 15]
	3:0	**CorrectPoints - Correct 1 (Encoder-only)** Format: U4 This field specifies the points used in the topmost RC region when sum_act > sum_max **Value** / **Name** [0, 15]
7	31:28	**CV7 - Clamp Value 7 (Encoder-only)** Exists If: //Encoder Only
	27:24	**CV6 - Clamp Value 6 (Encoder-only)** Exists If: //Encoder Only Format: U4
	23:20	**CV5 - Clamp Value 5 (Encoder-only)** Exists If: //Encoder Only Format: U4
	19:16	**CV4 - Clamp Value 4 (Encoder-only)** Exists If: //Encoder Only Format: U4
	15:12	**CV3 - Clamp Value 3 (Encoder-only)** Exists If: //Encoder Only Format: U4
	11:8	**CV2 - Clamp Value 2 (Encoder-only)** Exists If: //Encoder Only Format: U4
	7:4	**CV1 - Clamp Value 1 (Encoder-only)** Exists If: //Encoder Only Format: U4
	3:0	**CV0 - Clamp Value 0 (Encoder-only)** If the magnitude of coefficients at locations assigned with CV0 (mapping shown below) exceeds 2CV0-1, they are replaced with 2CV0-1. For coefficients at locations marked as 'none', no

clamping is performed. The following mappings are only applied to luma and chroma blocks\subblocks containing AC coefficients (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

MFD_AVC_BSD_OBJECT

Source: VideoCS

Length Bias: 2

Description
The MFD_AVC_BSD_OBJECT command is the only primitive command for the AVC Decoding Pipeline. The same command is used for both CABAC and CAVLD modes. The Slice Data portion of the bitstream is loaded as indirect data object. Before issuing a MFD_AVC_BSD_OBJECT command, all AVC states of the MFD Engine need to be valid. Therefore the commands used to set these states need to have been issued prior to the issue of a MFD_AVC_BSD_OBJECT command.
Context switch interrupt is not supported by this command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFD_AVC_BSD_OBJECT Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h AVC_DEC Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h Format: OpCode
	20:16	**SubOpcode B** Default Value: 8h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Total Length - 2 Value: 5h — Name: Excludes DWord (0,1) = 0005 **[Default]**
1	31:0	**Indirect BSD Data Length**

MFD_AVC_BSD_OBJECT

		Format: U32 This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. This field must have the same alignment as the Indirect Object Data Start Address. AVC Short Format : It is the length in bytes of the bitstream data for the current slice, including Slice Header + Slice Data + Emulation Prevention Bytes + any filling trailing zeros after the last MB. Hardware ignores the contents after the last non-zero byte. Trailing zero is allowed and handled correctly in both CABAC and CAVLC modes.
2	31:29	**Reserved** Format: MBZ
	28:0	**Indirect BSD Data Start Address** Format: U29 This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This pointer is relative to the **MFD Indirect Object Base Address**. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the AVC bitstream data in both CABAC/CAVLD Modes. In implementing a phantom slice at the end of a picture for automatic error concealment, this field should set to 0. It includes the NAL Header (the NAL Header does not need to perform EMU detection). For AVC Base Layer, it is a single byte. But for MVC, the NAL Header is 4 Bytes long. These NAL Header Unit must be passed to HW in the compressed bitstream buffer. **Value** / **Name** [0,512MB)
3..5	95:0	**Inline Data** Format: **Inline Data Description for MFD_AVC_BSD_Object** All the required Slice Header parameters and error handling settings are captured as InLine Data of the AVC_BSD_OBJECT command. It has a fixed size of 3 DWs. Its definition is described in the following section: Inline Data Description.
6	31:13	**Reserved** Format: MBZ
	12:9	**Reserved**
	8	**Reserved**
	7:0	**Reserved**

MFD_AVC_DPB_STATE

MFD_AVC_DPB_STATE

Source: VideoCS

Length Bias: 2

This is a frame level state command used only in AVC Short Slice Bitstream Format VLD mode. RefFrameList[16] of interface is replaced with intel Reference Picture Addresses[16] of MFX_PIPE_BUF_ADDR_STATE command. The LongTerm Picture flag indicator of all reference pictures are collected into LongTermPic_Flag[16]. FieldOrderCntList[16][2] and CurrFieldOrderCnt[2] of interface are replaced with intel POCList[34] of MFX_AVC_DIRECTMODE_STATE command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_MULTI_DW Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h AVC_DEC Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h Format: OpCode
	20:16	**SubOpcode B** Default Value: 6h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Total Length - 2 Value: 9h — Name: Excludes DWord (0,1) **[Default]**
1	31:16	**LongTermFrame_Flag[16][1 bit]** One-to-one correspondence with the entries of the Intel RefFrameList[16]. 1 bit per reference frame.

MFD_AVC_DPB_STATE

		Value / **Name** 1 — the picture is a long term reference picture 0 — the picture is a short term reference picture
	15:0	**Non-ExistingFrame_Flag[16][1 bit]** One-to-one correspondence with the entries of the Intel RefFrameList[16]. 1 bit per reference frame. **Value** / **Name** / **Description** 1 / INVALID / the reference picture in that entry of RefFrameList[] does not exist anymore. 0 / VALID / the reference picture in that entry of RefFrameList[] is a valid reference **Programming Notes** When an element of the list of frames is not relevant (e.g., due to the corresponding reference entry being empty or being marked as "not used for reference"), the value of the corresponding bit of NonExistingFrameFlags shall be set to 0.
2	31:0	**UsedForReference_Flag[16][2 bits]** One-to-one correspondence with the entries of the Intel RefFrameList[16]. 2 bits per reference frame. **Value** / **Name** / **Description** 0 / NOT_REFERENCE / indicates a frame is "not used for reference". 1 / TOP_FIELD / bit[0] indicates that the top field of a frame is marked as "used for reference". 2 / BOTTOM_FIELD / bit[1] indicates that the bottom field of a frame is marked as "used for reference". 3 / FRAME / bit[1:0] indicates that a frame (or field pair) is marked as "used for reference".
3..10	255:0	**LTSTFrameNumList[16][16 bits]** One-to-one correspondence with the entries of the Intel RefFrameList[16]. 16 bits per reference frame.Depending on the corresponding LongTermFrame_Flag[], the content of this field is interpreted differently. **Value** / **Name** / **Description** 1 / LongTermFrame_Flag[i] / LTSTFrameNumList[i] represent LongTermFrameIdx. 0 / ShortTermFrame_Flag[i] / LTSTFrameNumList[i]represent Short Term Picture FrameNum. **Programming Notes** When an element of the list of frames is not relevant (e.g., due to the corresponding reference entry being empty or being marked as "not used for reference"), the value of the LTSTFrameNumList entry shall be set to 0.
11..18	255:0	**ViewIDList[16][16 bits]**

		One-to-one correspondence with the entries of the Intel RefFrameList[16]. 16 bits per reference frame. The view ids are 10-bits, the upper 6 bits are ignored."000000" & ViewId1[9:0] & "000000" & ViewId0[9:0] **Programming Notes** When an Intel RefFrameList[i] is not an valid entries, Viewid should be set to 0x00
19..22	127:0	**ViewOrderListL0[16][8 bits]** One-to-one correspondence with the entries of the Intel RefFrameList[16]. 8 bits per reference frame. The view order need 4-bits, the upper 4 bits are ignored. 0000 & ViewOrder3[3:0] & 0000 & ViewOrder2[3:0] & 0000 & ViewOrder1[3:0] & 0000 & ViewOrder0[3:0] **Programming Notes** When the ViewOrderListL0[i] is not an valid inter-view reference, its corresponding ViewOrder should be set to 0xF Since only interview with the same polarity will be used, there is no need to have field bit in this list. Hardware is going to append correct polarity bit as needed.
23..26	127:0	**ViewOrderListL1[16][8 bits]** One-to-one correspondence with the entries of the Intel RefFrameList[16]. 8 bits per reference frame. The view order need 4-bits, the upper 4 bits are ignored. 0000 & ViewOrder3[3:0] & 0000 & ViewOrder2[3:0] & 0000 & ViewOrder1[3:0] & 0000 & ViewOrder0[3:0] **Programming Notes** When the ViewOrderListL1[i] is not an valid inter-view reference, its corresponding ViewOrder should be set to 0xF Since only interview with the same polarity will be used, there is no need to have field bit in this list. Hardware is going to append correct polarity bit as needed.

MFD_AVC_PICID_STATE

MFD_AVC_PICID_STATE

Source: VideoCS

Length Bias: 2

This is a frame level state command used for both AVC Long and Short Format in VLD mode.PictureID[16] contains the pictureID of each reference picture (16 maximum) so hardware can uniquely identify the reference picture across frames (this will be used for DMV operation).This command will be needed for both short and long format.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_MULTI_DW Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h MFD_AVC_DPB_STATE Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h DEC Format: OpCode
	20:16	**SubOpcode B** Default Value: 5h MEDIA_ Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 0008h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:1	**Reserved** Format: MBZ

<table>
<tr><td></td><td>0</td><td>PictureID Remapping Disable
<table><tr><td></td><td></td></tr></table>
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>AVC decoder will use 16 bits Picture ID to handle DMV and identify the reference picture</td><td>Desc</td></tr><tr><td>1h</td><td>AVC decoder will use 4 bits FrameStoreID (index to RefFrameList) to handle DMV and identify the reference picture (This causes DMV logic to function the same)</td><td>Desc</td></tr></table>
<table><tr><th>Programming Notes</th></tr><tr><td>If Picture ID Remapping Disable is "1", PictureIDList will not be used.</td></tr></table></td></tr>
<tr><td>2..9</td><td>255:0</td><td>PictureIDList[16][16 bits]
<table><tr><td></td><td></td></tr></table>
One-to-one correspondence with the entries of the Intel RefFrameList[16]. 16 bits per reference frame. PictureID of each Frame uniquely identifies the reference picture across frames. The same number cannot be reused until the reference picture is completely retired(no longer used for reference)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>

MFD_AVC_SLICEADDR

MFD_AVC_SLICEADDR

Source: VideoCS

Length Bias: 2

This is a Slice level command used only for 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 concealment should be invoked to generate those missing MBs. For AVC 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**

MFD_AVC_SLICEADDR

		Format: MBZ
	11:0	**DWord Length** Format: =n Total Length - 2 **Value** 2h — **Name** **[Default]**
1	31:0	**Indirect BSD Data Length** Format: U32 This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. Driver always programs this up to 5 bytes; for bitstream less than 5 bytes, driver program the lesser value. (Emulation Prevention Byte should never happen for the first 5 bytes when the max picture size can only be 4Kx4K)It is the length in bytes of the bitstream data for the current slice, including Slice Header + Slice Data + Emulation Prevention Bytes + any filling trailing zeros after the last MB. Hardware ignores the contents after the last non-zero byte. Trailing zero is allowed and handled correctly in both CABAC and CAVLC modes.
2	31:29	**Reserved** Format: MBZ
	28:0	**Indirect BSD Data Start Address** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This pointer is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the AVC bitstream data in both CABAC/CAVLD Modes. In implementing a phantom slice at the end of a picture for automatic error concealment, this field should set to 0.It includes the NAL Header Byte. (but does not perform EMU detection).Must provide a valid MB address, even if error. MB must be clamped to within a pic boundary. **Value** [0,512MB) — **Name**
3	31:13	**Reserved** Format: MBZ
	12:9	**Reserved**
	8	**AVC NAL Type First Byte Override Bit** Format: U1 This bit indicates hardware should use the NAL Type (provided below) programmed by driver

instead of using the one from bitstream. The NAL byte from bitstream will not be correct.

Value	Name	Description
0	Use Bitstream Decoded NAL Type	NAL Type should come from first byte of decoded bitstream.
1	Use Driver Programmed NAL Type	NAL Type should come from "Driver Provided NAL Type Values" programmed by driver.

7:0 **Driver Provided NAL Type Value**

Format:	U8

This will replace the first byte of the NAL unit, containing forbidden_zero_bit, nal_ref_idc, and nal_unit_type.

Programming Notes
This byte should be ignored if “AVC NAL Type First Byte Override Bit” is programmed to 0

MFD_IT_OBJECT

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** Format: =n Total Length - 2 Note: Regardless of the mode, inline data must be present in this command. The length excludes DWord (0,1)

Value	Name	Description
0Ch	AVC	There are total of 7 inline DWs for AVC (in addition to DW0-DW6 here)
10h	VC1	There are total of 11 inline DWs for VC1 (in addition to DW0-DW6 here)

MFD_IT_OBJECT

		0Bh | MPEG2 | There are total of 6 inline DWs for AVC (in addition to DW0-DW6 here)
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** Format: | MBZ
	28:0	**Indirect IT-MV Data Start Address Offset** This field specifies the memory starting address (offset) of the MV data to be fetched into the IT pipeline for processing. This pointer is relative to the Indirect IT-MV Object Base Address.Hardware ignores this field if indirect data is not present, i.e. the Indirect MV Data Length is set to 0. Alignment of this address depends on the mode of operation.AVC-IT Mode: It must be DWord aligned (since each MV is 4 bytes in size). This field is only valid in AVC decoder IT mode (VC1 and MPEG uses inline MV data). **Value** | **Name** [0,512MB) |
3	31:12	**Reserved** Format: | MBZ
	11:0	**Indirect IT-COEFF Data Length** | This field provides the length in bytes of the indirect data, which contains all the non-zero coefficients for the current MB. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect IT-COEFF Data Start Address field is ignored. Since each IT-COEFF data is 1 DW in size, with 12 bits, this field can be extended to support up to 4:4:4 format.(256 pixel * 3 byte pixel components * 4 bytes per coeff).This field must be integer multiple of 16-bytes for AVC (since each coefficient is 4 bytes in size).This field is only valid in AVC, VC1, MPEG2 decoder IT mode. **Value** | **Name** [0,3072] | In bytes [0, 256*3*4]
4	31:29	**Reserved**

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

MFD_IT_OBJECT		
AVC--There are 7 addition DWs so n = 13 VC1--There are 11 addition DWs so n = 17 MPEG2--There are 6 addition DWs so n = 12		format. VC1-IT Modes: Hardware interprets this data in the specified format. MV inline MPEG2-IT Modes: Hardware interprets this data in the specified format. (IS mode) MV inline For AVC there 7 DWords of inline data, hence N is equal to 13.

MFD_JPEG_BSD_OBJECT

MFD_JPEG_BSD_OBJECT

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.

MFD_JPEG_BSD_OBJECT

2 — 31:29 Reserved

Format:	MBZ

2 — 28:0 Indirect Data Start Address

Format:	IndirectObjectOffset[28:0]

This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This pointer is relative to the BSD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the JPEG bitstream data

3 — 31:29 Reserved

Format:	MBZ

3 — 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.

3 — 15:13 Reserved

Format:	MBZ

3 — 12:0 Scan Vertical Position

Format:	U13 bits in blocks

This field indicates the vertical position (in block units) of the first MCU in the Scan.

4 — 31 Reserved

Format:	MBZ

4 — 30 Interleaved

Value	Name	Description
0	Non-Interleaved	one component in the Scan
1	Interleaved	multiple components in the Scan

4 — 29:27 Scan Components

Bit0: Y Bit1: U Bit2: V For example, if non-interleaved Y, then it will be set to 001b. If interleaved Y, U, and V, it will be set to 111b.

4 — 26 Reserved

Format:	MBZ

	25:0	**MCU Count** Format: U26 This field indicates the number of MCUs in the Scan.
5	31:16	**Reserved** Format: MBZ
	15:0	**RestartInterval(16 bit)** Format: U16 Specifies the number of MCU in restart interval. Valid values are 1->0xFFFFValue of 0 implies that all the SCAN have only one ECS.

MFD_MPEG2_BSD_OBJECT

MFD_MPEG2_BSD_OBJECT

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**

MFD_MPEG2_BSD_OBJECT

		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 possible to have a slice data with large length (including zero-padding bytes). As the data beyond 0x10E00 would only be zero bytes for a valid slice data zero-padding restriction is removed
2	31:29	**Reserved** Format: MBZ
	28:0	**Indirect Data Start Address** Format: IndirectObjectOffset[28:0] This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This pointer is relative to the BSD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the MPEG2 VLD bitstream data This address points to the first byte of the MB layer data, i.e. not including slice header.
3..4	63:0	**Inline Data** Format: **MFD_MPEG2_BSD_OBJECT Inline Data Description** All the required Slice Header parameters and error handling settings are captured as MPEG2_BSD_OBJECT Inline Data Descriptor structures. It has a fixed size of 2 DWs. Its definition is described in the next section.

MFD_VC1_BSD_OBJECT

MFD_VC1_BSD_OBJECT

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

MFD_VC1_BSD_OBJECT

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.GEN6 Long Format : It is the length in bytes of the bitstream data for the current slice/picture. It includes the first byte of the first macroblock and the last byte of the last macroblock in the slice/picture. Specifically, the zero-padding bytes (if present) and the next start-code are excluded. Hardware ignores the contents after the last non-zero byte (trailing zeros). This field is sized to support VC1 AP@L4 Level bitstream. It includes the byte that contains the First MB Bit OffsetGEN7 Short Format : It is the length in bytes of the bitstream data for the current slice, including Picture/Slice Header + Emulation Prevention Bytes + any filling trailing zeros after the last MB. Hardware ignores the contents after the last non-zero byte. Trailing zero is allowed and handled correctly.
2	31:29	**Reserved** Format: MBZ
	28:0	**Indirect Data Start Address** Format: IndirectObjectOffset[28:0] This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This pointer is relative to the MFD Indirect Object Base Address.Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VC1 bitstream data. **Value**: [0,512MB) — **Name**:
3	31:24	**Reserved** Format: MBZ
	23:16	**Slice Start Vertical Position** This field specifies the position in y-direction of the first macroblock in the Slice in unit of macroblocks. For SecondField this value is reset to zero as opposed to the VC1 spec Ref: 9.1.2 Slice Layer. This field is for both Long and Short VC1 Interface Format.
	15:9	**Reserved** Format: MBZ
	8:0	**Next Slice Vertical Position**

MFD_VC1_BSD_OBJECT

		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.
4	31:16	**First_MB_Byte_Offset of Slice Data or Slice Header** For DXVA2 VC1 Short Format only. It gives the byte offset to locate the first MB data in the bitstream for a slice, relative to the Indirect BSD Data Start Address.
	15:5	**Reserved** Format: MBZ
	4	**Emulation Prevention Byte Present** 0h – H/W needs to perform Emulation Byte Removal 1h – H/W does not need to perform Emulation Byte Removal
	3	**Reserved** Format: MBZ
	2:0	**FirstMbBitOffset (First Macroblock Bit Offset)** Format: U3 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.

MFD_VC1_LONG_PIC_STATE

MFD_VC1_LONG_PIC_STATE

Source: VideoCS

Length Bias: 2

MFX_VC1_LONG PIC_STATE command encapsulates the decoding parameters that are read or derived from bitstream syntax elements above (inclusive) picture header layer. These parameters are static for a picture and when slice structure is present, these parameters are not changed from slice to slice of the same picture. Hence, this command is only issued at the beginning of processing a new picture and prior to the VC1_*_OBJECT command. The values set for these state variables are retained internally across slices. Only the parameters needed by hardware (BSD unit) to decode bit sequence for the macroblocks in a picture layer or a slice layer are presented in this command. Other parameters such as the ones used for inverse transform or motion compensation are provided in MFX_VC1_PRED_PIPE_STATE command. This Long interface format is intel proprietary interface. Driver will need to perform addition operations to generate all the fields in this command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFD_VC1_LONG_PIC_STATE Format: OpCode
	26:24	**Media Command Opcode** Default Value: 2h VC1_DEC Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h Format: OpCode
	20:16	**SubOpcode B** Default Value: 1h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 0004h Excludes DWord (0,1) Format: =n Total Length - 2

MFD_VC1_LONG_PIC_STATE

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
[0,254]	Value_0_to_254	a valid range of [0,254] [1, 255] MB
255	Value_255	

Programming Notes

Note: Even though the Advanced Profile allows frame dimensions (width, height) to not be aligned to macroblock boundary, 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
[0,254]	Value_0_to_254	[1,255] MB
255	Value_255	

2 — 31:24 — Bitplane Buffer Pitch Minus 1

Format:	U8-1 Pitch in (Bytes - 1).

Specifies the bitplane buffer pitch in (#Bytes - 1). Bitplane buffer is a linear buffer. It is needed only when the bitplane is not encoded as raw, and therefore is present in the header explicitly. In VC1 Long Format (Gen6 and Gen7), it is written by an application and later read by the HW. But in VC1 Short Format (Gen7 only), it is written and read by H/W only. This field is specified for better performance.

Value	Name
[0,255]	

Programming Notes

For Gen6 : The pitch must be equal to PictureWidthInMBs/2.For Gen7 VC1 Long Format : The pitch must be equal to PictureWidthInMBs/2.For Gen7 VC1 Short Format : If Pic Width is less than or equal to 2K pixels, bitplane pitch is set to 64 (one cacheline; programmed as 63) bytes per MB row. If Pic Width is greater than 2K pixels, bitplane pitch is set to 128 (two cachelines; programmed as 127) bytes per MB row. This field is not used in IT mode, used in VLD mode only. For VC1 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).When the current picture being decoded is an I, P or BI, this bit is set to 0, since there is no DMV read in these picture decoding process. This field is not used in IT mode, used in VLD mode only.

14 **ImplicitQuantizer**

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

13 **Interpolation Rounder Control**

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 PictureParameters data structure

12 **SyncMarker**

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

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

Programming Notes
This field is only valid in VLD mode. For Simple Profile, SyncMarker must set to 0.For Main Profile, SyncMarker can be set to 0 or 1.This field is used in both intel and MS 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 Spec, now I have fixed it to match with 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 PictureParameters data structure. Bit 11 of Motion Vector Mode = 1 for Luma Bilinear MC; = 0 for Luma Bicubic MCBit 8 of Motion Vector Mode = 1 for half-sample Chroma motion = 0 for quarter-sample Chroma motion. This field is used in both VLD and IT modes.

7 **RangeReductionScale**

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

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

Programming Notes
This bit is derived by driver for Main Profile only. Ignored in Simple and Advanced Profiles. This field is used in both VLD and IT modes. This is derived by driver from the history of RANGERED and RANGEREDFRM syntax elements (i.e. of forward/preceding reference picture) and those of the current picture. RANGERED is the same as (bPicOverflowBlocks » 3) & 1. RANGEREDFRM is

the same as (bPicDeblocked » 5) & 1. For the current picture is a B picture, this field represents the state of the forward/preceding reference picture onlyDriver is responsible to keep RangeReductionScale, RangeReduction Enable and RANGERED Present Flag of current picture coherent.

6 **RangeReduction Enable**

This field specifies whether on-the-fly pixel value range reduction should be performed for the preceding (or forward) reference picture. Along with RangeReductionScale to specify whether scale up or down should be performed. It is not the same value as RANGEREDFRM Syntax Element (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 ON at the picture level. This 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 intel VC1 VLD mode only, not in 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 reserved When FCM = 10 | 11 (a Field Picture)000 = I/I001 = I/P010 = P/I011 = P/P100 = B/B101 = B/BI110 = BI/B111 = BI/BIAlthough, for a field picture, it is set for a field-pair, but HW will only look at one field state only, and the other field state is don't care. This field is read and qualified with the SecondField flag internally. This field is unique to intel VC1 VLD Long format, and is used in IT mode as well. For VC1 IT mode, driver needs to convert the interface to intel HW VLD Long Format interface.
	25:24	**FCM (Frame Coding Mode)** This is the same as the variable FCM defined in VC1.This field must be set to 0 for Simple and Main Profiles. This field is unique to intel VC1 VLD Long format, and is used in IT mode as well. For VC1 IT mode, driver needs to convert the interface to intel HW VLD Long Format interface.

MFD_VC1_LONG_PIC_STATE

		Value / **Name** / **Description** 00b / Disable / Progressive Frame Picture 01b / Enable / Interlaced Frame Picture 10b / / Field Picture with Top Field First 11b / / Field Picture with Bottom Field First
	23:21	**Reserved** Format: MBZ
	20:16	**AltPQuant (Alternative Picture Quantization Value)** This field is identical to the variable ALTPQUANT which is derived from VOPDQUANT configuration in the VC1 standard. This field must be set to 0 for Simple/Main I and BI pictures as VOPDQUANT is not present. This field is used in intel VC1 VLD Long Format mode only, not used in VC1 VLD and IT modes.
	15:13	**Reserved** Format: MBZ
	12:8	**PQuant (Picture Quantization Value)** Format: U5 This is the same as the calculated variable PQUANT in VC1 standard where PQuant = PQINDEX, except when QUANTIZER = 0 and PQINDEX > 8, it is given asPQuant = (PQINDEX < 29) ? PQINDEX - 3 : PQINDEX*2 - 31This field is used in all picture types (I, P, B and BI) and all operating modes (IT mode and intel and VLD modes).
	7:0	**BScaleFactor** BScaleFactorThis field is the scale factor for computing Direct-mode motion vectors. It is derived from the variable BFRACTION in the VC1 standard, section 8.4.5.4.There are only 21 valid values corresponding to the 21 encodings of BFRACTION as shown in the table here. Other values are reserved.MSB of this field can be used to determine if BFRACTION is greater than or equal to 1/2, which is used to determine Motion Prediction Type for B pictures. Effectively, condition "BFRACTION >= 1/2" is equivalent to condition "BScaleFactor >= 128".This field is only valid for B pictures. This field is used only in intel VC1 VLD Long format mode, it is not used in VC1 VLD and IT modes. BFRACTION VLCBFRACTIONBScaleFactor0001/21280011/3850102/31700111/4641003/41921011/5511102/510211100003/515311100014/520411100101/64311100115/621511101001/73711101012/77411101103/711111101114/714811110005/718511110016/722211110101/83211110113/89611111005/816011111017/8224
4	31:30	**Reserved** Format: MBZ
	29:28	**UnifiedMvMode (Unified Motion Vector Mode)**

This field is a combination of the variables MVMODE and MVMODE2 in the VC1 standard, for parsing Luma MVD from the bitstream. This field is used to signal 1MV vs 4MVallowed (Mixed Mode). This field is also used to signal Q-pel or Half-pel MVD read from the bitstream. The bicubic or bilinear Luma MC interpolation mode is duplicate information from Motion Vector Mode field, and is ignored here. This field is used in intel VC1 VLD Long Format mode only, it is not used in VC1 VLD and IT modes.

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

27 **FourMvSwitch (Four Motion Vector Switch)**

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

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

24 **NumRef (Number of References)**

This field indicates how many reference fields are referenced by the current (field) picture. It is

identical to the variable NUMREF in the VC1 standard. This field is only valid for field P picture (FCM = 10 | 11).This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.

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

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

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

19:16 **FwdRefDist (Reference Distance)**

Format:	U4

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

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

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

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

MFD_VC1_LONG_PIC_STATE

	7:4	**AltPQuantEdgeMask (Alternative Picture Quantization Edge Mask)** This field is a bit mask for the four edges in clock-wise order, indicating whether AltPQuant is used for the edge macroblocks.It is derived based on the following variables DQUANT, DQUANTFRM, DQPROFILE, DQSBEDGE, DQDBEDGE, and DQBILEVEL defined in the VC1 standard, as shown in Error! Reference source not found..This field is valid only if AltPQuantConfig is 01. Bit 0: Left picture edge macroblocksBit 1: Top picture edge macroblocksBit 2: Right picture edge macroblocksBit 3: Bottom picture edge macroblocksThis field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.
	3:2	**AltPQuantConfig (Alternative Picture Quantization Configuration)** This field specifies the way AltPQuant is used in the picture. It determines how to compute the macroblock quantizer step size, MQUANT. It is derived based on the following variables DQUANT, DQUANTFRM, DQPROFILE, DQSBEDGE, DQDBEDGE, and DQBILEVEL defined in the VC1 standard, as shown in Error! Reference source not found.. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and modes.

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

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

Value	Name	Description
0h		Non-uniform
1h		Uniform

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

Value	Name	Description
0h		bitplane buffer is not present

1h		bitplane buffer is present	

30 **ForwardMbRaw**

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

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

29 **MvTypeMbRaw**

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

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

27 **DirectMbRaw**

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

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

26 **OverflagsRaw**

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

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

25 **AcPredRaw**

This field indicates whether the ACPRED field is coded in raw or non-raw mode. This field is only valid when PictureType is I or BI. This field is unique to intel VC1 VLD Long format mode, and is

not used in IT and VC1 modes.

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

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

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

23	**Reserved**

Format:	MBZ

22:20	**MvTab (Motion Vector Table)**

Format:	U3

This field specifies which motion vector table(s) is (are) used for motion vector (differential) decoding in a P or B picture. This field is the combination of the variables MVTAB and IMVTAB in the VC1 standard. Two bits are defined for progressive frame pictures; And two or three bits are defined for interlaced field/frame pictures depending on NUMREF and P or B picture types. This field is valid for P and B pictures. It is not valid for I pictures. For P or B progressive frame pictures0 = Motion Vector Differential VLD Table 01 = Motion Vector Differential VLD Table 12 = Motion Vector Differential VLD Table 23 = Motion Vector Differential VLD Table 3The other encodings are reserved. For 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 reserved. For 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 reserved. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.

19:18	**FourMvBpTab (4-MV Block Pattern Table)** This field specifies which table is used to decode the 4-MV block pattern (4MVBP) syntax element in 4-MV macroblocks. It is identical to the variables 4MVBPTAB in the VC1 standard, section 9.1.1.37. This field is valid only in interlace frame P, B pictures, or interlace field P, B pictures. It is not valid for I picture. For interlace field P and B pictures, it is only valid if UnifiedMvMode is equal to Mixed-MV Type. For interlace frame P picture, it is only valid if FourMvSwitch is 1.For interlace frame B picture, it is always valid.0 = 4MVBP Table 01 = 4MVBP Table 12 = 4MVBP Table 23 = 4MVBP Table 3This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.

	17:16	**TwoMvBpTab (2MV Block Pattern Table)** This field specifies which table is used to decode the 2MV block pattern (2MVBP) syntax element in 2MV field macroblocks. It is identical to the variables 2MVBPTAB in the VC1 standard, section 9.1.1.36. This field is valid only in interlace frame P/B pictures. It is not valid for I picture, nor for interlace field P or B pictures.0 = 2MVBP Table 01 = 2MVBP Table 12 = 2MVBP Table 23 = 2MVBP Table 3This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.
	15:14	**Reserved** Format: MBZ
	13:12	**TransType (Picture-level Transform Type)** Format: U2 This field specifies the Transform Type at picture level. It is identical to the variable TTFRM in the VC1 standard, section 7.1.1.41.This field is only valid when TransTypeMbFlag is 1. Otherwise, it is reserved and MBZ. This field is set to 00 when VSTRANSFORM is 0 in the entry point layer.00 = 8x8 Transform01 = 8x4 Transform10 = 4x8 Transform11 = 4x4 Transform. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.
	11	**TransTypeMbFlag (Macroblock Transform Type Flag)** This field indicates whether Transform Type is fixed at picture level or variable at macroblock level. It is identical to the variable TTMBF in the VC1 standard, section 7.1.1.40.This field is set to 1 when VSTRANSFORM is 0 in the entry point layer. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.

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

	10:8	**MbModeTab (Macroblock Mode Table)** This field signals which code table is used to decode the macroblock mode syntax element (MBMODE) in the macroblock layer in a P or B picture. This field is identical to the variables MBMODETAB in the VC1 standard, section 9.1.1.33. This field is valid for interlace frame P, B picture and interlace field P, B picture. It is not valid for I picture, nor progressive frame P, B pictures.Two bits are defined for interlace frame P, B pictures; And three bits are defined for interlaced field P, B pictures.Two bits are defined for interlace frame P, B pictures. There are two set of code tables selected based on if UnifiedMvMode is equal to 4-MV Type or not. 0 = Code Table 01 = Code Table 12 = Code Table 23 = Code Table 3Other encodings are invalidThree bits are defined for interlace field P, B pictures. There are two set of code tables selected based on if UnifiedMvMode is equal to Mixed-MV Type or not. 0 = Code Table 01 = Code Table 12 = Code Table 23 = Code Table 34 = Code Table 45 = Code Table 56 = Code Table 67 = Code Table 7This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.
	7:6	**TransAcY (Picture-level Transform Luma AC Coding Set Index, TRANSACTABLE2** BitFieldDesc

MFD_VC1_LONG_PIC_STATE

	5:4	**TransAcUV (Picture-level Transform Chroma AC Coding Set Index, TRANSACTABLE)** This field, together with PQINDEX, specifies which intra AC coding set to be used for decoding the non-zero AC coefficients in a coded luma (Y) block. This field is the combination of the variables TRANSACFRM and TRANSACFRM2 in the VC1 standard. For I pictures, TransAcY is the same as TRANSACFRM2. For other pictures, it is the same as TRANSACFRM, and therefore must be programmed to be the same as TransAcUV. This field is valid for all picture types.0 = Coding set index 01 = Coding set index 12 = Coding set index 23 is invalid This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.
	3	**TransDcTab (Intra Transform DC Table)** This field specifies whether the low motion tables or the high motion tables are used to decode the Transform DC coefficients in intra-coded blocks. This field is identical to the variable TRANSDCTAB in the VC1 standard, section 8.1.1.2.This field is valid for all picture types .This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.
	2:0	**CbpTab (Coded Block Pattern Table)** This field specifies the table used to decode the CBPCY syntax element for each coded macroblock in P and B pictures. This field is combination of the variable CBPTAB for P and B frame pictures and the variable ICBPTAB in interlace field P, B pictures and interlace frame P, B pictures in the VC1 standard (Table 52 and Table 102).This field is reserved and MBZ for I or BI pictures as I only has a fixed table.000 = Table 0 (Table 169 for P, B frames or Table 124 otherwise)001 = Table 1 (Table 170 for P, B frames or Table 125 otherwise)010 = Table 2 (Table 171 for P, B frames or Table 126 otherwise)011 = Table 3 (Table 172 for P, B frames or Table 127 otherwise)100 = Table 4 (Table 128 for interlace field/frame P, B pictures)101 = Table 5 (Table 129 for interlace field/frame P, B pictures)110 = Table 6 (Table 130 for interlace field/frame P, B pictures)111 = Table 7 (Table 131 for interlace field/frame P, B pictures)This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.

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

MFD_VC1_SHORT_PIC_STATE

MFD_VC1_SHORT_PIC_STATE

Source: VideoCS

Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFD_VC1_SHORT_PIC_STATE Format: OpCode
	26:24	**Media Command Opcode** Default Value: 2h VC1_DEC Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h Format: OpCode
	20:16	**SubOpcode B** Default Value: 0h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 0003h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:24	**Reserved** Format: MBZ
	23:16	**Picture Height** Format: U8-1 Picture Height in Macroblocks This field indicates the height of the picture in unit of macroblocks. For example, for a 1920x1080 frame picture, PictureHeightInMBs equals 68 (1080 divided by 16, and rounded up,

MFD_VC1_SHORT_PIC_STATE

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

		either half-pel or quarter-pel precision.0XX0 = Chroma Quarter -pel + Luma bicubic. (can only be 1MV)0XX1 = Chroma Half-pel + Luma bicubic. (can be 1MV or 4MV)1XX0 = Chroma Quarter -pel + Luma bilinear. (can only be 1MV)1XX1 = Chroma Half-pel + Luma bilinearNote: Bits 19:16 are taken from bMVprecisionAndChromaRelation in PictureParameters data structure.Bit 19 of Motion Vector Mode = 1 for Luma Bilinear MC; = 0 for Luma Bicubic MCBit 16 of Motion Vector Mode = 1 for half-sample Chroma motion = 0 for quarter-sample Chroma motion.This field is used in both VLD and IT modes. Before the polarity of Chroma Half-pel or Q-pel is reversed from Spec.
	15	**DmvSurfaceValid** Indicated when the DMV read surface is valid. This surface stored the direct motion vectors. This field is set to B pictures that can refer to a previous P picture for DMV. If there is an I-picture before a B (in decoding order) then this field is not set (as a result, zero's DMV's will be assumed while decoding the B picture. That is, there is no explicit DMV buffer for an I-picture). This field is not used in IT mode, used in VLD mode only.
	14:12	**Reserved** Format: MBZ
	11	**VC1 Profile** specifies the bitstream profile. Note: This is required because 128 is added for intra blocks post inverse transform in advanced profile and also to find out if Motion vectors are adjusted or not. This field is used in both VLD and IT modes.
	10:6	**Reserved** Format: MBZ
	5	**Backward Prediction Present Flag** Note : a B picture that only uses forward prediction may have this flag set to 1 as well. Driver may still need to provide a valid reference picture index. This field is used in both VC1 VLD mode and IT mode. It is the same parameter as bPicBackwardPrediction in 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 VC1 VLD and IT mode.
	4	**Intra Picture Flag** This field is used in both VC1 VLD mode and IT mode. It is the same parameter as bPicIntra in 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 VC1 VLD

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

and IT mode.

Value	Name	Description
0h		entire picture can have a mixture of intra and inter MB type or just inter MB type.
1h		entire picture is coded in intra MB type

	3	**SecondField** This flag is set for the second field in field pictures. This field is used in both VLD and IT modes.
	2	**Reserved** Format: MBZ
	1:0	**Picture Structure** This field is used in both VC1 VLD mode and IT mode. It is the same parameter as bPicStructure in VC1 spec. The Picture Structure and Progressive Pic Type are used to derive the picture structure as specified in FCM, in VC1 VLD and IT mode.
3	31	**Reserved** Format: MBZ
	30	**Overlap Smoothing Enable Flag** This field is the decoded syntax element OVERLAP in bitstreamIndicates if Overlap smoothing is ON at the picture levelThis field is used in both VLD and IT modes
	29	**Range Reduction Scale** Access: None This field specifies whether the reference picture pixel values should be scaled up or scaled down on-the-fly, if RangeReduction is Enabled.NOTE: This bit is derived by driver for Main Profile only. Ignored in Simple and Advanced Profiles. This field is used in both VLD and IT modes. This is derived by driver from the history of RANGERED and RANGEREDFRM syntax elements (i.e. of forward/preceding reference picture) and those of the current picture. RANGERED is the same as (bPicOverflowBlocks » 3) & 1. RANGEREDFRM is the same as (bPicDeblocked » 5) & 1. For the current picture is a B picture, this field represents the state of the forward/preceding reference picture onlyDriver is responsible to keep RangeReductionScale, RangeReduction Enable and

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

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

RANGERED Present Flag of current picture coherent.

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

28 **Range Reduction Enable**

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

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

27:24 **Reserved**

Format:	MBZ

23:22 **Progressive Pic Type**

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

MFD_VC1_SHORT_PIC_STATE

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 VC1 VLD mode only, not used in IT and intel VC1 VLD Long Format modes.

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

15:14 QUANTIZER

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

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

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

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

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

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

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

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

10:8 MAXBFRAMES

Number of consecutive B Frames.

7 PANSCAN Present Flag

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

6 REFDIST_FLAG

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

5 **LOOPFILTER Enable Flag**

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

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

4 **FastUVMCFlag (Fast UV Motion Compensation Flag)**

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

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

3 **EXTENDED_MV Present Flag**

BitFieldDesc

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

2:1 **DQUANT**

Access:	None
Format:	U2

Use for Picture Header Parsing of VOPDUANT elements

Value	Name	Description
0h	**[Default]**	
00b		no VOPDQUANT elements; Quantizer cannot vary in frame, same quantization step size PQUANT is used for all MBs in the frame
01b		refer to VC1 Spec. for all the MB position dependent quantizer selection

		10b | | The macroblocks located on the picture edge boundary shall be quantized with ALTPQUANT while the rest of the macroblocks shall be quantized with PQUANT. 11b | Reserved |
	0	**VSTRANSFORM flag** Value: 0h | Name: Disable | Description: variable-sized transform coding is not enabled Value: 1h | Name: Enable | Description: variable-sized transform coding is enabled
4	31:29	**Reserved** Format: MBZ (for possible future change to BFraction Enumeration)
	28:24	**BFraction Enumeration** This field is the scale factor for computing Direct-mode motion vectors. It is derived from the variable BFRACTION in the VC1 standard, section 8.4.5.4.There are only 21 valid values corresponding to the 21 encodings of BFRACTION as shown in the table here. Other values are reserved. The VLD decoded value of BFRACTION (from the picture header) is mapped into an enum value from 0 to 20. This field is only valid for B pictures. This field is used only in VC1 VLD mode, it is not used in Intel VC1 VLD Long Format mode and IT mode. BFRACTION VLCBFRACTION Enum0001/200011/310102/320111/431003/441011/551102/5611100003/5711100014/5811100101/6911100115/61011101001/71111101012/71211101103/71311101114/71411110005/71511110016/71611110101/81711110113/81811111005/81911111017/8201111111BI Pic Indicator31 (optional)
	23	**Reserved** Format: MBZ Advanced Profile only; RANGE_MAPY_FLAG Range Mapping not supported
	22:20	**Reserved** Format: MBZ Advanced Profile only; RANGE_MAPY Range Mapping not supported
	19	**Reserved** Format: MBZ Advanced Profile only; RANGE_MAPUV_FLAG Range Mapping not supported
	18:16	**Reserved** Format: MBZ Advanced Profile only; RANGE_MAPUV Range Mapping not supported
	15:9	**Reserved** Format: MBZ

	8	**4MV Allowed Flag**
	7	**POSTPROC Flag**
	6	**PULLDOWN**
	5	**INTERLACE**
	4	**TFCNTRFLAG**
	3	**FINTERFLAG**
	2	**REFPIC Flag** For a BI picture, REFPIC flag must set to 0For I and P picture, REFPIC flag must set to 0.For a B picture, REFPIC flag must set to 0, except for a B-field in interlaced field mode which can be 0 or 1 (e.g. the top B field can be used as a reference for decoding its corresponding bottom B-field in a field pair).In VLD mode, this flag cannot be used as an optimization signaling for an I or P picture that is not used as a reference picture. This field is used in both VC1 VLD mode and IT mode. It is the same parameter as bPicDeblockConfined[bit2] in 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 VC1 VLD and IT mode. **Value** / **Name** / **Description** 0h / / the current picture after decoded, will never used as a reference picture 1h / / the current picture after decoded, will be used as a reference picture later
	1	**PSF**
	0	**EXTENDED_DMV Present Flag** **Value** / **Name** / **Description** 0h / **[Default]** / Extended_DMV is not present in the picture header 1h / / Extended_DMV is present in the picture header

MFD_VP8_BSD_OBJECT

MFD_VP8_BSD_OBJECT

Source: VideoCS

Length Bias: 2

The MFD_VP8_BSD_OBJECT command is the only primitive command for the VP8 Decoding Pipeline. The Partitions of the bitstream is loaded as indirect data object. Before issuing a MFD_VP8_BSD_OBJECT command, all VP8 frame level states of the MFD Engine need to be valid. Therefore the commands used to set these states need to have been issued prior to the issue of a MFD_VP8_BSD_OBJECT command. Context switch interrupt is not supported by this command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFD_VP8_BSD_OBJECT Format: OpCode
	26:24	**Media Command OpCode** Default Value: 4h VP8_DEC Format: OpCode
	23:21	**subOpcodeA** Default Value: 1h Format: OpCode
	20:16	**subOpcodeB** Default Value: 8h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 14h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:21	**Reserved** Format: MBZ
	20:16	**Partition0 CPBAC Entropy Count** Pass the Partition0 CPBAC State to HW. Max value is 24.
	15:8	**Partition0 CPBAC Entropy Range**

		Pass the Partition0 CPBAC State to HW.
	7:6	**Reserved** Format: MBZ
	5:4	**Coded Num of Coeff Token Partitions** Num of Partitions = 2^CodedNumCoeffTokenPariitons. 0 = 1 Partition only 1 = 2 Partitions 2 = 4 Partitions 3 = 8 Partitions are present in the bitstream.
	3	**Reserved** Format: MBZ
	2:0	**Partition0 FirstMBBitOffset from Frame Header** Allow HW to jump to the location in the bitstream where per MB information starts in the Partition0.
2	31:24	**Partition0 CPBAC Entropy Value** Pass the Partition0 CPBAC State to HW.
	23:0	**Reserved** Format: MBZ
3	31:24	**Reserved** Format: MBZ
	23:0	**Indirect Partition0 Data Length** This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition. **Programming Notes** This needs to be set to the (actual Partition 0 length + 1) in bytes
4	31:0	**Indirect Partition0 Data Start Offset** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.
5	31:24	**Reserved** Format: MBZ
	23:0	**Indirect Partition1 Data Length** This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition. **Programming Notes**

MFD_VP8_BSD_OBJECT

		This needs to be set to the (actual Partition 1 length + 1) in bytes
6	31:0	**Indirect Partition1 Data Start Offset** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.
7	31:24	**Reserved** Format: MBZ
	23:0	**Indirect Partition2 Data Length** This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition. **Programming Notes** This needs to be set to the (actual Partition 2 length + 1) in bytes
8	31:0	**Indirect Partition2 Data Start Offset** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.
9	31:24	**Reserved** Format: MBZ
	23:0	**Indirect Partition3 Data Length** This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition. **Programming Notes** This needs to be set to the (actual Partition 3 length + 1) in bytes
10	31:0	**Indirect Partition3 Data Start Offset** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.
11	31:24	**Reserved** Format: MBZ
	23:0	**Indirect Partition4 Data Length** This field provides the length in bytes of the indirect data. A value zero indicates that indirect

		data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition. **Programming Notes** This needs to be set to the (actual Partition 4 length + 1) in bytes
12	31:0	**Indirect Partition4 Data Start Offset** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.
13	31:24	**Reserved** Format: MBZ
	23:0	**Indirect Partition5 Data Length** This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition. **Programming Notes** This needs to be set to the (actual Partition 5 length + 1) in bytes
14	31:0	**Indirect Partition5 Data Start Offset** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.
15	31:24	**Reserved** Format: MBZ
	23:0	**Indirect Partition6 Data Length** This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition. **Programming Notes** This needs to be set to the (actual Partition 6 length + 1) in bytes
16	31:0	**Indirect Partition6 Data Start Offset** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.

MFD_VP8_BSD_OBJECT

17	31:24	**Reserved** Format: MBZ
	23:0	**Indirect Partition7 Data Length** This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition. **Programming Notes** This needs to be set to the (actual Partition 7 length + 1) in bytes
18	31:0	**Indirect Partition7 Data Start Offset** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.
19	31:24	**Reserved** Format: MBZ
	23:0	**Indirect Partition8 Data Length** This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition. **Programming Notes** This needs to be set to the (actual Partition 8 length + 1) in bytes
20	31:0	**Indirect Partition8 Data Start Offset** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.
21	31	**Concealment Method** This field specifies the method used for concealment when error is detected. **Value** / **Name** / **Description** 0 / Intra 16x16 Prediction / A copy from the current picture is performed using Intra 16x16 Prediction method. 1 / Inter P Copy / A copy from collocated macroblock location is performed from the concealment reference indicated by the ConCeal_Pic_Id field.
	30:18	**Reserved** Format: MBZ
	17:16	**Conceal_Pic_Id (Concealment Picture ID)**

		Exists If: [Concealment Method] == 1 This field identifies the picture in the reference list to be used for concealment. This field is only valid if Concealment Method is Inter P Copy. 00 - Last Decoded Picture 01 - Golden Reference Picture 02 - Alternate Reference Picture 03 - User provided Reference Picture
	15	**Reserved** Format: MBZ
	14	**BSDPrematureComplete Error Handling** It occurs in situation where the decode is completed but there are still data in the bitstream. **Value** / **Name** 0 – Ignore the error and continue (masked the interrupt), assume the hardware automatically perform the error handling 1 – Set the interrupt to the driver (provide MMIO registers for MB address R/W)
	13	**Reserved** Format: MBZ
	12	**MPR Error (MV out of range) Handling** **Value** / **Name** 0 – Ignore the error and continue (masked the interrupt), assume the hardware automatically perform the error handling 1 – Set the interrupt to the driver (provide MMIO registers for MB address R/W)
	11	**Reserved** Format: MBZ
	10	**Entropy Error Handling** **Value** / **Name** 0 – Ignore the error and continue (masked the interrupt), assume the hardware automatically perform the error handling 1 – Set the interrupt to the driver (provide MMIO registers for MB address R/W)
	9	**Reserved** Format: MBZ
	8	**MB Header Error Handling** **Value** / **Name** 0 – Ignore the error and continue (masked the interrupt), assume the hardware automatically perform the error handling 1 – Set the interrupt to the driver (provide MMIO registers for MB address R/W)
	7:0	**Reserved** Format: MBZ

MFX_AVC_DIRECTMODE_STATE

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 AVC Interface. The DMV buffers are not required to be programmed for encoder mode.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_SINGLE_DW Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h AVC_COMMON Format: OpCode
	23:21	**SubOpcodeA** Default Value: 0h Format: OpCode
	20:16	**SubOpcodeB** Default Value: 2h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 0045h Excludes DWord (0,1) Format: =n Total Length - 2
1..32	1023:0	**Direct MV Buffer for Reference Frame 0 to 15 - Base Address** Format: **SplitBaseAddress64ByteAligned[16]** This field is for the Pre-Deblocking Destination Address, and provides the base address of the DMV buffer for reference frames 2 to 31. They are needed if the current B-Picture has MBs

		coded in direct mode. This 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]). This field is changed to one per frame: both top and bottom field share the same Direct MV Buffer Base Address. **Programming Notes** This field is ignored if PreDeblockOutEnable is set to 0 (disable).
33	31:0	**Direct MV Buffer for Reference Frame 0 to 15 - Attributes** Format: **MemoryAddressAttributes**
34..35	63:0	**Direct MV Buffer for Write - Base Address** Format: **SplitBaseAddress64ByteAligned** 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. GraphicsAddress is a 64-bit value [63:0], but only a portion of it is used by hardware. The upper reserved bits are ignored and MBZ. Some GraphicsAddress fields only specify the upper address bits. For example, GraphicsAddress[47:12] is a 4KB page address.
36	31:0	**Direct MV Buffer for Write - Attributes** Format: **MemoryAddressAttributes**
37..70	1087:0	**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 Current Frames/Fields There are 34 POC entries in the list. For reference picture, only the lower 32 POC [0-31] entries can be used, and POCList[ ] is indexed by the frame_store_ID[4:0], which is obtained from RefPicList L0/L1[RefPicIdx]. frame_Store_IDbit[0] (indicator for Top/Bottiom Field). For current picture, all 34 POC entries [0-33] can be addressed by POCList[img_dec_fs_idc[4:0]«1 + img_structure[1]]. For frame-only mode, every other entry is skipped. For MBAFF and field-only picture, each entry is a field POC, and every two entries are paired.

MFX_AVC_IMG_STATE

MFX_AVC_IMG_STATE

Source: VideoCS

Length Bias: 2

This must be the very first command to issue after the surface state, the pipe select and base address setting commands. This command supports both Long and Short VLD and IT AVC Decoding Interface.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_AVC_IMG_STATE Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h AVC_COMMON Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 0h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 0Ch Excludes DWord (0,1) Format: =n 00Eh, used for normal decode and encode mode000h, a special case to provide a dummy image state for stitch mode operation. In this case, fields in DW1 which is part of the dummy image state command are ignored by hardware.
1	31:16	**Reserved** Format: MBZ
	15:0	**Frame Size** Format: U16-1 in MB unit

The value for FrameSizeInMBs must match the product of FrameWidthInMBs and FrameHeightInMBs, Max.
Screen resolution is therefore limited to 256 x 256 in MB unit. It is enough to cover all the Profile-Level specified in the current HD-DVD specification. E.g.. for 1920x1080, FrameSizeInMBs[15:0] = 8160 (1920/16 * 1088/16; rounded up 1080.) This parameter is specified for Intel interface only.

Value	Name	Description
[0,16383]		representing Number of MBs [1,16384]

2 — 31:24 — Reserved

Format:	MBZ

(bit[31:24] must be zero to match the 16-bit definition for FrameHeightInMBsMinus1)

2 — 23:16 — Frame Height

Format:	U8-1 in MB unit

It is set to the value of (FrameHeightInMBsMinus1+ 1). Since the max value for FrameHeightInMBs is 255, the max allowed value for FrameHeightInMBsMinus1 is only 254. The min value for FrameHeightInMBs is 1.Although the max. value that can be specified for FrameHeightInMBs is 255 (in the current implementation), FrameWidthInMBs * FrameHeightInMBs must not exceed the max value of FrameSizeInMBs[14:0].e.g. for 1920x1080, FrameHeightInMBs[7:0] is equal to 68 (1080 divided by 16, and rounded up, i.e. effectively specified as 1088 instead).It is derived from FrameHeightInMbs = (2 - frame_mbs_only_flag) * PicHeightInMapUnits and PicHeightInMbs = FrameHeightInMbs / (1 + field_pic_flag) internally done. For MBAFF, PicHeightInMapUnits is in MB pair unit, so the bitstream sends only half frame height.

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

2 — 15:8 — Reserved

Format:	MBZ

(bit[15:8] must be zero to match the 16-bit definition for FrameWidthInMBsMinus1)

2 — 7:0 — Frame Width

Format:	U8-1 in MB unit

It is set to the value of (FrameWidthInMBsMinus1+ 1). Since the max value for FrameWidthInMBs is 255, the max allowed value for FrameWidthInMBsMinus1 is only 254. The min value for FrameWidthInMBs is 1.Although the max. value that can be specified for FrameWidthInMBs is 255 (in the current implementation), FrameWidthInMBs * FrameWidthInMBs must not exceed the max value of FrameSizeInMBs[14:0].e.g. for 1920x1080, FrameHeightInMBs[7:0] is equal to 68 (1080 divided by 16, and rounded up, i.e. effectively specified as 1088 instead).It is derived from FrameWidthInMbs = (2 - frame_mbs_only_flag) * PicWidthInMapUnits and PicWidthInMbs = FrameWidthInMbs / (1 + field_pic_flag) internally done. For MBAFF, PicWidthInMapUnits is in MB pair unit, so the bitstream sends only half frame width.

		Value / **Name** / **Description** [0,255] / / representing width [1,256]
3	31:29	**Reserved** Format: MBZ (bit[31:29] must be zero to match the 8-bit definition for InitQpChroma[1])
	28:24	**Second Chroma QP Offset** Signed integer value. It should be in the range of -12 to +12 (according to AVC spec).It specifies the offset for determining QP Cr from QP Y. It is set to the upper 5 bits of the value of the syntax element (Chroma_qp_offset[9:0]) read from the current active PPS.Chroma_qp_offset [4:0] - chroma_qp_offset_bits (from the current active PPS)Chroma_qp_offset [9:5] - second_chroma_qp_offset_bits
	23:21	**Reserved** Format: MBZ (bit[23:21] must be zero to match the 8-bit definition for InitQpChroma[1])
	20:16	**First Chroma QP Offset** Signed integer value. It should be in the range of -12 to +12 (according to AVC spec).It specifies the offset for determining QP Cb from QP Y. It is set to the lower 5 bits of the value of the syntax element (Chroma_qp_offset[9:0]) read from the current active PPS.Chroma_qp_offset [4:0] - chroma_qp_offset_bits (from the current active PPS)Chroma_qp_offset [9:5] - second_chroma_qp_offset_bits
	15:14	**Reserved** Format: MBZ
	13	**RhoDomain Rate Control Enable** Format: Enable This field indicates if RhoDomain related parameters are present in the MFX_AVC_IMAGE_STATE. (AverageMacroblockQP). It enables the Rho Domain statistics collection. **Value** / **Name** / **Description** 0 / Disable / RhoDomain rate control parameters are not present in MFX_AVC_IMAGE_STATE 1 / Enable / RhoDomain rate control parameters are present in MFX_AVC_IMAGE_STATE. **Programming Notes** This field must set to '0' for B pictures.
	12	**Weighted_Pred_Flag** Format: Enable

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

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

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

11:10 **Weighted_BiPred_Idc**

(This field is follows strictly AVC interface.)

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

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

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

The current encoding picture structure can only takes on 3 possible values

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

Programming Notes
img_structure[0] can be used as a flag to distinguish between frame and field structure. It must be consistent with the field_pic_flag setting in the Slice Header. This parameter is specified for Intel interface only, as a separate state (instead the img_structure[1] is embedded inside the picture definition).

7:0 **Reserved**

Format:	MBZ

4 31:16 **MinFrameWSize**

Default Value:	0h

Format:	U16

Minimum Frame Size [15:0] (in Word, 16-bit)(Encoder Only) Minimum Frame Size is specified to compensate for intel Rate Control Currently zero fill (no need to perform emulation byte insertion) is done only to the end of the CABAC_ZERO_WORD insertion (if any) at the last slice of a picture. Intel encoder parameter. The caller should always make sure that the value, represented by Minimum Frame Size, is always less than maximum frame size **FrameBitRateMax (DWORD 10 bits** 29:16). This field is reserved in Decode mode.
The programmable range 0...2^18-1
When MinFrameWSizeUnits is 00.
Programmable range is 0...2^20-1 when MinFrameWSizeUnits is 01.
Programmable range is 0...2^26-1 when MinFrameWSizeUnits is 10.
Programmable range is 0...2^32-1 when MinFrameWSizeUnits is 11.

15 **MbStatEnabled**

Format:	Enable

Enable reading in MB status buffer (a.k.a. encoding stream-out buffer) Note: For multi-pass encoder, all passes except the first one need to set this value to 1. By setting the first pass to 0, it does save some memory bandwidth.
In VDenc mode this must be set to zero as no MB level rate control is used.

Value	Name	Description
0	Disable	Disable Reading of Macroblock Status Buffer
1	Enable	Enable Reading of Macroblock Status Buffer

14 **LoadSlicePointerFlag**

Format:	Enable

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

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

13 **Reserved**

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

Value	Name	Description

0	PACKED	use packed MV format
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	Description
0	CAVLC bit-serial encoding mode	Desc
1	CABAC bit-serial encoding mode.	Desc

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

6 **ImgDisposableFlag**

Current Img Disposable Flag or Non-Reference Picture Flag

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

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

5 **ConstrainedIPredFlag**

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

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 picture. It is set to the value of the syntax element in the current active PPS.

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

	2	**FrameMbOnlyFlag** Frame MB only flag, frame_mbs_only_flagIt is set to the value of the syntax element in the current active SPS. **Value / Name / Description** 0 / FALSE / not true ; effectively enables the possibility of MBAFF mode. 1 / TRUE / true, only frame MBs can occur in this sequence, hence disallows the MBAFF mode and field picture.
	1	**MbaffFlameFlag** MBAFF mode is active, mbaff_frame_flag.It is derived from MbaffFrameFlag = (mb_adaptive_frame_field_flag && ! field_pic_flag). mb_adaptive_frame_field_flag is a syntax element in the current active SPS and field_pic_flag is a syntax element in the current Slice Header. They both are present only if frame_mbs_only_flag is 0. Although mbaff_frame_flag is a Slice Header parameter, its value is expected to be the same for all the slices of a picture.It must be consistent with the mb_adaptive_frame_field_flag, the field_pic_flag and the frame_mbs_only_flag settings.This bit is valid only when the img_structure[1:0] indicates the current picture is a frame. **Value / Name / Description** 0 / FALSE / not in MBAFF mode 1 / TRUE / in MBAFF mode
	0	**FieldPicFlag** Field picture flag, field_pic_flag, specifies the current slice is a coded field or not.It is set to the same value as the syntax element in the Slice Header. It must be consistent with the img_structure[1:0] and the frame_mbs_only_flag settings.Although field_pic_flag is a Slice Header parameter, its value is expected to be the same for all the slices of a picture. **Value / Name / Description** 0h / FRAME / a slice of a coded frame 1h / FIELD / a slice of a coded field
5 [ExistsIf]Encode Only	31	**Trellis Quantization Enabled (TQEnb)** Format: Enable The TQ improves output video quality of AVC CABAC encoder by selecting quantized values for each non-zero coefficient so as to minimize the total R-D cost. This flag is only valid AVC CABAC mode. Otherwise, this flag should be disabled. **Value / Name / Description** 0h / Disable / Use Normal 1h / Enable / Use Trellis quantization In VDENC mode, Enable Trellis Quantization in MFX. VDENC can control trellis Quantization based on it's programming.
	30:28	**Trellis Quantization Rounding (TQR)** This rounding scheme is only applied to the quantized coefficients ranging from 0 to 1

when TQEnb is set to 1 in AVC CABAC mode. One of the following values is added to quantized coefficients before truncating fractional part.

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

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

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

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

26:17 **Reserved**

Format:	MBZ

16 **NonFirstPassFlag**

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

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

15:13 **Reserved**

Format:	MBZ

12 **Reserved**

Format:	MBZ

11:10 **MinFrameWSizeUnits**

This field is the Minimum Frame Size Units

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

11b	16Kb	Minimum Frame Size is in 16Kbytes

9 MbRateCtrlFlag - MB level Rate Control Enabling Flag

MB Rate Control conformance mask

In VDenc mode, this field must be zero as frame level rate control is used.

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

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

8 Reserved

Format:	MBZ

7 Intra/InterMbIpcmFlag - ForceIPCMControlMask

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

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

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

6:4 Reserved

Format:	MBZ

3 FrameSzUnderFlag - FrameBitRateMinReportMask

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

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

2 FrameSzOverFlag - FrameBitRateMaxReportMask

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

Value	Name	Description

		0 Disable Do not update bit0 of MFC_IMAGE_STATUS control register. 1 Enable Set bit0 and bit 1 of MFC_IMAGE_STATUS control register if the total frame level bit counter is greater than or equal to Frame Bit rate Maximum limit.
	1	**InterMbMaxBitFlag - InterMBMaxSizeReportMask** This is a mask bit controlling if the condition of any inter MB in the frame exceeds InterMBMaxSize. **Value / Name / Description** 0 Disable Do not update bit0 of MFC_IMAGE_STATUS control register. 1 Enable Set bit0 of MFC_IMAGE_STATUS control register if the total bit counter for the current MB is greater than the Inter MB Conformance Max size limit.
	0	**IntraMbMaxBitFlag - IntraMBMaxSizeReportMask** This is a mask bit controlling if the condition of any intra MB in the frame exceeds IntraMBMaxSize. **Value / Name / Description** 0h Disable Do not update bit0 of MFC_IMAGE_STATUS control register. 1 Enable set bit0 of MFC_IMAGE_STATUS control register if the total bit counter for the current MB is greater than the Intra MB Conformance Max size limit.
6 [ExistsIf]Encode Only	31:28	**Reserved**
	27:16	**InterMbMaxSz** Format: U12 This field, Inter MB Conformance Max size limit,indicates the allowed max bit count size for Inter MB
	15:12	**Reserved** Format: MBZ
	11:0	**IntraMbMaxSz** Exists If: //Intra Only Format: U12 This field, Intra MB Conformance Max size limit,indicates the allowed max bit count size for Intra MB All IPCM MBs should ignore this Max size limit.
7	31:0	**Reserved** Format: MBZ
8	31:24	**SliceDeltaQpMax[3]**

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

		Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP for total bit-count below FrameBitRateMin - first 1/8 regionThis field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is less than FrameBitRateMin and greater than or equal to 1/8 the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of [(FrameBitRateMin-FrameBitRateMinDelta»3), FrameBitRateMin).
	23:16	**SliceDeltaQpMin[2]** Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP for bit-count below FrameBitRateMin - below 1/ 8 and above 1/ 4This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between one-eighth and quarter the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of [(FrameBitRateMin- FrameBitRateMinDelta»2), (FrameBitRateMin- FrameBitRateMinDelta»3)).
	15:8	**SliceDeltaQpMin[1]** Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP for bit-count below FrameBitRateMin- below 1/4 and above 1/ 2This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between quarter and half the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of [(FrameBitRateMin- FrameBitRateMinDelta»1), (FrameBitRateMin- FrameBitRateMinDelta»2)).
	7:0	**SliceDeltaQpMin[0]** Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice Level Delta QP for bit-count below FrameBitRateMin - below 1/ 2This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is below FrameBitRateMin by more than half the distance of FrameBitRateMinDelta , i.e., in the range of [0, (FrameBitRateMin- FrameBitRateMinDelta»1).
10 [ExistsIf]Encode Only	31	**FrameBitrateMaxUnit** This field is the Frame Bitrate Maximum Limit Units. **Value** / **Name** / **Description** 0 / Byte / FrameBitRateMax is in units of 32 Bytes when FrameBitrateMaxUnitMode is 1 and in units of 128 Bytes if

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

30 **FrameBitrateMaxUnitMode**

This field is the Frame Bitrate Maximum Limit Units.

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

29:16 **FrameBitRateMax**

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

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

15 **FrameBitrateMinUnit**

This field is the Frame Bitrate Minimum Limit Units.

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

14 **FrameBitrateMinUnitMode**

This field is the Frame Bitrate Minimum Limit Units.

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

13:0 **FrameBitRateMin**

RangeThe programmable range 0-512KB When FrameBitrateMinUnit is in 0.Programmable range is 0-8190 KB when FrameBitrateMinUnit is in 1.This field is the Frame Bitrate Minimum Limit ()This field along with FrameBitrateMinUnit determines minimum allowed bits in a Frame before Multi-Pass gets triggered (when enabled). In other words, multi-pass is triggered when the actual frame byte count is less than this

		value. When FrameBitrateMinUnitMode is 0 (compatibility mode) bits 0:11 should be used, bits 12 and 13 should be 0.
11 [ExistsIf]Encode Only	31	**Slice Stats Streamout Enable** **Description** Enable signal for Slice size streamout. When this bit is set, the slice size is written to SliceSize StreamOut Data (MFX_PIPE_BUF_ADDR_STATE DW 65,66,67) using one word. The slice size is in units of Bytes.
	30:16	**FrameBitRateMaxDelta** Format: U15 This field is used to select the slice delta QP when FrameBitRateMax Is exceeded. It shares the same FrameBitrateMaxUnit. When FrameBitrateMaxUnitMode is 0(compatibility mode) bits 16:27 should be used, bits 28, 29 and 30 should be 0. **Value** / **Name** / **Description** 0-1024KB / / The Programmable range 0-1024KB when FrameBitRateMaxUnit is 0. 0-16380KB / / The Programmable range is 0-16380KB when FrameBitRateMaxUnit is 1. 0h / **[Default]** /
	15	**Reserved** Format: MBZ
	14:0	**FrameBitRateMinDelta** Range: The programmable range 0-1024KB When FrameBitrateMinUnit is in 32Bytes.Programmable range is 0-16380KB when FrameBitrateMinUnit is in 4Kbytes. This field is used to select the slice delta QP when FrameBitRateMin Is exceeded. It shares the same FrameBitrateMinUnit. When FrameBitrateMinUnitMode is 0(compatibility mode) bits 0:11 should be used, bits 12, 13 and 14 should be 0.Note: HW requires the following condition FrameBitRateMinDelta <= 2*FrameBitRateMinMust be true, otherwise it may cause unpredicted behavior.
12	31:0	**Reserved** Format: MBZ
13	31:30	**Reserved** Format: MBZ
	29	**Current Picture Has Performed MMCO5** Set to 1 if the current Pic has performed the memory_management_control_operation = = 5.
	28:24	**Number of Reference Frames** Format: U5

		Range: Range 0 to MaxDpbSize (=16 for Level 4.1) Specifies the maximum number of reference frames (frames, field pairs, unpaired field) existed in the current DBP for decoding the current picture.
	23:22	**Reserved** Format: MBZ
	21:16	**Number of Active Reference Pictures from L1** Format: U6-1 Specifies the initial maximum reference index value minus 1 to access the L1 Reference List. It is extracted from PPS. It corresponds to the number of active reference pictures from L1 to decode the current picture. It can be modified by the slice header if num_ref_idx_active_override_flag is set. Only valid for B picture. Value: [0,31] — Name:
	15:14	**Reserved** Format: MBZ
	13:8	**Number of Active Reference Pictures from L0** Format: U6-1 Specifies the initial maximum reference index value minus 1 to access the L0 Reference List. It is extracted from PPS. It corresponds to the number of active reference pictures from L0 to decode the current picture. It can be modified by the slice header if num_ref_idx_active_override_flag is set. Valid for both P and B pictures. Value: [0,31] — Name:
	7:0	**Initial QP Value** Format: S7 Range: [-26,25] Initial QP value for a Slice, extracted from PPS. It may further get modified by slice_qp_delta in slice header and mb_qp_delta in MB header.
14 [ExistsIf] Short Format only	31:24	**Log2_max_pic_order_cnt_lsb_minus4** Exists If: //Short Format Only It is a SPS syntax element, used to determine how many bits in the bitstream are used to represent pic_order_cnt_lsb syntax element in the slice header. Unsigned
	23:16	**Log2_max_frame_num_minus4** Exists If: //Short Format Only It is a SPS syntax element, used to determine how many bits in the bitstream are used to represent frame_num syntax element in the slice header. Unsigned.

MFX_AVC_IMG_STATE

	15	**deblocking_filter_control_present_flag** Exists If: //Short Format Only It is a PPS syntax element, indicates if more deblocking filter control syntax elements are present in the slice header.
	14:12	**num_slice_groups_minus1** Exists If: //Short Format Only BitField It is a PPS syntax element. Use for Slice Header parsing only, to read in slice_group_change_cycle, if any, but is not used by H/W, i.e. no slice group support.Desc
	11	**redundant_pic_cnt_present_flag** Exists If: //Short Format Only It is a PPS syntax element. Use for Slice Header parsing only, to read-in redundant_pic_cnt, if any, but is not used by H/W, i.e. no support for redundant slice processing.
	10:8	**slice_group_map_type** Exists If: //Short Format Only It is a PPS syntax element. Use for Slice Header parsing only, to read in slice_group_change_cycle, if any, but is not used by H/W, i.e. no slice group support.
	7:4	**Reserved** Format: MBZ 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	31:16	**Curr Pic Frame Num**

[ExistsIf] Short Format only		Exists If: //Short Format Only Format: U16 Derived from Slice Header syntax element
	15:0	**Slice Group Change Rate** Exists If: //Short Format Only Format: U16-1 It is a PPS syntax element Use for Slice Header parsing only, to read in slice_group_change_cycle, if any, but is not used by H/W, i.e. no slice group support.
16 [ExistsIf]: Short Format only	31	**Inter View Order Disable** Exists If: //Short Format Only It indicates how to append inter-view picture into initial sorted reference list. (due to ambiguity in the MVC Spec) **Value** / **Name** / **Description** 0h / Default **[Default]** / View Order Ascending 1h / Disable / View ID Ascending
	30:22	**Reserved** Format: MBZ
	21:18	**Max View IDXL1** Exists If: //Short Format Only It is a PPS syntax element corresponding to Anchor/Non-Anchor Reference ListL1 It indicates the maximum number of inter-view picture for Reference List L1
	17:16	**Reserved** Format: MBZ
	15:12	**Max View IDXL0** Exists If: //Short Format Only Reference ListL0 It indicates the maximum number of inter-view picture for Reference List L0
	11:10	**Reserved**

		Format: MBZ
	9:0	**Current Frame View ID** Exists If: //Short Format Only It indicates the View ID of the current decoding frame
17	31:22	**Reserved** Format: MBZ
	21:16	**RhoDomain AverageMacroblockQP** Exists If: //[RhoDomain Rate Control] == 1 Format: U6
	15:9	**Reserved** Format: MBZ
	8	**Extended RhoDomain Statistics Enable** This parameter enables PAK to generate RhoDomain statistics from marcoblock QP and fractional QP computation. **Value** / **Name** / **Description** 0 / Disable / RhoDomain statistics generated from Frame Qp and no fractional QP computation. 1 / Enable / Enable MB QP based RhoDomain statistics and fractional QP computation. Note: By default, this is always enabled on. Only the enable case will be validated.
	7:6	**Reserved** Format: MBZ
	5:3	**Fractional QP offset** Format: U3 Start position from the top of the Frame where increased Quantization parameter is added.
	2:0	**Fractional QP input**

<table>
<tr><th colspan="3">MFX_AVC_IMG_STATE</th></tr>
<tr><td></td><td></td><td>In a set of 8 rows, Qp is incremented by 1 for F_qp rows.</td></tr>
<tr><td>18</td><td>31:0</td><td>Reserved<table><tr><td></td><td></td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>19</td><td>31:0</td><td>Threshold Size in Bytes<table><tr><td></td><td></td></tr><tr><td>Format:</td><td>U32</td></tr></table>When a slice exceeds this value in bytes, hardware will end current slice as soon as possible and insert a new slice boundary. Note there is no guarantee that the actual slice size will meet this value, it is a hint to the HW to end the slice as soon as possible (which could be 2-5 macroblocks in the future from this detection point).</td></tr>
<tr><td>20</td><td>31:0</td><td>Target Slice Size in Bytes<table><tr><td></td><td></td></tr><tr><td>Format:</td><td>U32</td></tr></table>In VDENC mode, this field indicates the target slice size in Bytes for slice size conformance feature. When the actual slice size exceeds "Target slice size in Bytes", HW sets "VDENC Slice Overflow Error" bit in MFC_IMAGE_STATUS_CONTROL register.</td></tr>
</table>

MFX_AVC_REF_IDX_STATE

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 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

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 picture index and intel frame store ID. As such, the final RefPicList L0/L1[ ] that the driver passes onto the H/W is not the same as that defined.

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

<table>
<tr><th colspan="3">MFX_AVC_REF_IDX_STATE</th></tr>
<tr><td></td><td>15:12</td><td>Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11:0</td><td>DWord Length
<table><tr><td>Default Value:</td><td>0008h</td></tr><tr><td>Format:</td><td>=n</td></tr></table>Excludes DWords 0,1</td></tr>
<tr><td>1</td><td>31:1</td><td>Reserved
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>0</td><td>RefPicList Select
Num_ref_idx_l1_active is resulted from the specifications in both PPS and Slice Header for the current slice. However, since the full reference list L0 and/or L1 are always sent, only present flags are specified instead.
This parameter is specified for Intel interface only.
<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
Programming Notes:
In VDENC mode, we support only three forward reference pictures.
HW supports only 1:1 reference index to reference picture mapping.
Reference index 0 should point to Reference picture 0, whose address is specified inMFX_PIPE_BUF_ADDR DW 19,20 (The reference picture numbers may be different from bit-stream reference picture)
Reference index1 should point to Reference picture2, whose address is specified inMFX_PIPE_BUF_ADDR 23, 24 (The reference picture numbers may be different from bit-stream reference picture)
Reference index2 should point to Reference picture4, whose address is</td><td>255:0</td><td>Reference List Entry
This set of fields is always present whenever this command is issued.
It always specifies the full 32 reference pictures in the selected list, regardless they are "existing picture" or not. If a picture is non-existing, the corresponding entry should be set to all ones. Each list entry is 1 byte. A 32-bit DW can hold 4 list entries in the following format
31:24 entry X+3 (e.g. listY_3)
23:16 entry X+2 (e.g. listY_2)
15:8 entry X+1 (e.g. listY_1)
7:0 entry X (e.g. listY_0)
X is replaced by the paddr[2:0] * 4 ; paddr[5:0] with 0x20 and 0x27, and Y is replaced by 0 or 1. The byte definition for a reference picture :
Bit 7 : Non-Existing - indicates that frame store index that should have been at this entry did not exist and was replaced by an index 0 (a valid entry) for error</td></tr>
</table>

MFX_AVC_REF_IDX_STATE		
specified inMFX_PIPE_BUF_ADDR 27,28 (The reference picture numbers may be different from bit-stream reference picture)		concealment • Bit 6 : Long term bit - set this reference picture to be used as long term reference • Bit 5 : Field picture flag - indicates frame/field • Bit 4:0 : Frame store index or Frame Store ID (Bit 4:1 is used to form the binding table index in intel implementation) This is the final Reference List L0 or L1 after any reordering specified in the Slice Header as well as modified by the driver, and its indices values are all translated to the intel specification. If the reference picture is a frame (Bit5 = 1), frame store ID is always an even number. This list is used in outputting MV information by the BSD unit in VLD mode. DMV access also reads and writes Mvlist0 using this frame store ID. If this set of fields is interpreted as Reference Index Mapping Table L0/L1, the same field alignment is followed, i.e. 4 mapping entries per DW. Each mapping entry is one byte in size, but only the least significant 5 bits [4:0] is relevant. Driver should zero all the upper bits [7:5] for each entry.

MFX_AVC_SLICE_STATE

MFX_AVC_SLICE_STATE

Source: VideoCS

Length Bias: 2

Description

This is a slice level command and can be issued multiple times within a picture that is comprised of multiple slices. The same command is used for AVC encoder (PAK mode) and decoder (VLD and IT modes).

In VDEnc mode, this command is programmed for every super-slice. However not all parameters are allowed to change across super-slices.

Programming Notes

MFX_AVC_SLICE_STATE command is not issued for AVC Short Format Bitstream decode, instead MFD_AVC_SLICEADDR command is executed to retrieve the next slice MB Start Address X and Y by H/W itself.

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

MFX_AVC_SLICE_STATE

1	31:4	**Reserved** Format: MBZ
	3:0	**Slice Type** It is set to the value of the syntax element read from the Slice Header. Value / Name: 0000b – P Slice 0001b – B Slice 0010b – I Slice 0011b-1111b – Reserved **Programming Notes** Bits[3:2] must be 0
2	31:30	**Reserved** Format: MBZ
	29:24	**Number of Reference Pictures in Inter-prediction List 1** Format: U6 This field is valid only for encoding a B Slice, for which it is expected to have at least one entry in the reference list L1; otherwise (if Slice Type is not a B Slice), this field must be set to 0. This field can be derived for a B Slice from the Slice Header syntax element NumRefIdxActiveMinus1 as, Num_Ref_Idx_L1 = NumRefIdxActiveMinus1[1] + 1. Value / Name: 0-32
	23:22	**Reserved** Format: MBZ
	21:16	**Number of Reference Pictures in Inter-prediction List 0** Format: U6 This field is valid for encoding a P or B Slice, for which it is expected to have at least one entry in the reference list L0; otherwise (if Slice Type is not a P or B Slice), this field must be set to 0. This field can be derived for a P or B Slice from the Slice Header syntax element NumRefIdxActiveMinus1 as, Num_Ref_Idx_L0 = NumRefIdxActiveMinus1[0] + 1. Value / Name: 0-32
	15:11	**Reserved** Format: MBZ
	10:8	**Log 2 Weight Denom Chroma**

		Format: U3 Value / Name: 0-7
	7:3	**Reserved** Format: MBZ
	2:0	**Log 2 Weight Denom Luma** Format: U3 Description: 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(). In VDEnc Mode, this value is programmed per super slice. Value / Name: 0-7
3	31:30	**Weighted Prediction Indicator** This field indicates the Weighted Prediction mode for a P or B Slice. It is a combined field corresponding to the syntax element WeightedBiPredIdc or WeightedPredFlag read from the current active PPS. • If it is a B-Slice, these bits are interpreted as: 00b - Specifies the default weighted inter-prediction to be applied 01b - Specifies the explicit weighted inter-prediction to be applied 10b - Specifies the implicit weighted inter-prediction to be applied 11b - Reserved (not allowed) • If it is a P Slice, these bits are interpreted as: 00b - Disables weighted inter-prediction (Default weighted) 01b - Enables weighted inter-prediction (Explicit weighted) 10b - 11b - Reserved **Programming Notes** Only when in B Slice with Weighted_Pred_Idc = 1 (explicit weighted prediction), will there be a L1 and/or a L0 weight+offset tables being sent to the BSD unit through the Slice_State command. Only when in P Slice with Weighted_Pred_Idc = 1, will there be a L0 weight+offset table being sent to the BSD. If Weighted_Pred_Idc != 1 for B Slice or Weighted_Pred_Idc =0 for P Slice, no Slice_State command should be issued to send these tables. If still being issued, the data is read but ignored. DXVA specifies Weighted_Bipred and Weighted_Pred in frame-level state. However, these two flags are combined and specified in slice level for both P and B slice type.
	29	**Direct Prediction Type** Type of direct prediction used for B Slices. This field is valid only for Slice_Type = B Slice;

otherwise, it must be set to 0.

Value	Name
0	Temporal
1	Spatial

28:27 **Disable Deblocking Filter Indicator**

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

26 **Reserved**

Format:	MBZ

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

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

Value	Name
0-2	

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

23:22 **Reserved**

Format:	MBZ

21:16 **Slice Quantization Parameter**

Quantization Parameter for current slice. Derived from PPS and slice_delta_qp syntax element in Slice Header. It is needed for CABAC context initialization and deblocking filter control. And it is also used as the starting QP value in the very first MB of a slice. It is in the range of unsigned integer 0 to 51, for 8-bit pixel bit-depth.

15:12 **Reserved**

Format:	MBZ

11:8 **Slice Beta Offset Div2**

Format:	S3 2's Complement

Range: [-6, 6] Inclusive
Specifies the offset used in accessing the deblocking filter strength tables.

7:4 **Reserved**

		Format: MBZ
	3:0	**Slice Alpha C0 Offset Div2** Format: S3 2's Complement Range: [-6, 6] Inclusive Specifies the offset used in accessing the deblocking filter strength tables.
4	31:24	**Slice Vertical Position** This field specifies the position in y-direction of the first macroblock in the Slice in unit of macroblocks. The fields (Slice_MB_Start_Hor_Pos, Slice_MB_Start_Vert_Pos) are valid in VLD (decoding) mode only. They are ignored by hardware in decoding IT mode and encoding mode (whereas the position is provided by the per-macroblock object command). Derived **Programming Notes** Error Handling: Driver needs to check if FirstMbY starts at 0 on the first slice of frame. If not, driver needs to add a phantom slice with FirstMbX and FirstMbY set to 0.
	23:16	**Slice Horizontal Position** This field specifies the position in x-direction of the first macroblock in the Slice in unit of macroblocks. Derived **Programming Notes** Error Handling: Driver needs to check if FirstMbY starts at 0 on the first slice of frame. If not, driver needs to add a phantom slice with FirstMbX and FirstMbY set to 0.
	15	**Reserved** Format: MBZ
	14:0	**Slice Start Mb Num** Exists If: //Decoder Only The MB number (linear MB address in a picture) at the start of a Slice, it must match with the Slice Horizontal Position (Slice_MB_Start_Hor_Pos) and Vertical Position (Slice_MB_Start_Vert_Pos) in the picture. **Programming Notes** In creating the Phantom Slice for error concealment, this field should set to the total number of MB in the current picture + 1.
5	31:24	**Reserved** Format: MBZ
	23:16	**Next Slice Vertical Position** This field specifies the position in y-direction of the first macroblock in the next Slice in unit of macroblocks. This field is primarily used for error concealment. In the case that current slice is the last slice, this field should set to the height of picture (since y-direction is zero-based numbering).
	15:8	**Reserved** Format: MBZ

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

Value	Name
0	QP Panic
1	CBP Panic

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

21 **MB Type Direct Conversion Disable**

Exists If:	//B-Slice

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

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

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

20 **MB Type Skip Conversion Disable**

Exists If:	//P-Slice or B-Slice

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

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

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

19 **Is Last Slice**

Description
It is used by the zero filling in the Minimum Frame Size test.
Set this only for the last slice group

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

18 **Reserved**

17 **Header Insertion Present in Bitstream**

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

Programming Notes
Note: In VDEnc mode, the slice header PAK object maximum size is 25 DWs.
This need to be set only for super slice0.

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.

Programming Notes
This bit should be set for all super-slices.

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.

Programming Notes
This bit should only be set for the last super slice. In VDENC mode, SW should insert 1000 **VD_PIPELINE_FLUSH commands with VDENC_pipeline_Done set to 1 before inserting tail command. This is for delaying the tail insertion in HW. The HW recommendation is to insert tail only at the end of sequence to avoid performance loss since this restriction potentially cause performance degradation.**

14 **Reserved**

Format:	MBZ

13 **EmulationByteSliceInsertEnable**

To have PAK outputting SODB or EBSP to the output bitstream buffer

Value	Name	Description
0		outputting RBSP
1		outputting EBSP

12 **CabacZeroWordInsertionEnable**

To pad the end of a SliceLayer RBSP to meet the encoded size requirement.

Value	Name	Description
0		No Cabac_Zero_Word Insertion
1		Allow internal Cabac_Zero_Word generation and append to the end of RBSP (effectively can be used as an indicator for last slice of a picture, if the assumption is only the last slice of a picture needs to insert CABAC_ZERO_WORDs.

11:8 **Reserved**

Format:	MBZ

7:4 **Slice ID [3:0]**

To identify the output data (coding information record) returned for rate control from PAK to ENC and VPP.

3:2 **Reserved**

Format:	MBZ

1:0 **Stream ID [1:0]**

To identify the output data (coding information record) returned for rate control from PAK to ENC and VPP.

7 Encoder Only

31:29 **Reserved**

Format:	MBZ

28:0 **Indirect PAK-BSE Data Start Address (Write)**

Exists If:	//AVC Encode Mode

This field specifies the memory starting address (offset) to write out the compressed bitstream data from the BSE processing. This pointer is relative to the MFC Indirect PAK-BSE Object Base Address. It is a byte-aligned address for the AVC bitstream data in both CABAC/CAVLC Modes. For Write, there is no need to have a data length field. It is assumed the global memory bound check specified in the IND_OBJ_BASE_ADDRESS command (Indirect PAK-BSE Object Access Upper Bound) will take care of any illegal write access.

Value	Name
[0h,1FFFFFFFh]	

8 Encoder Only

31:24 **Magnitude of QP Max Negative Modifier**

Format:	U8

This field specifies the lower limit of the QP modifier.

Value	Name
0-51	

23:16 **Magnitude of QP Max Positive Modifier**

Format:	U8

This field specifies the upper limit of the QP modifier.

Value	Name

		0 - 15
	15:12	**Shrink Param - Shrink Resistance** Format: U4 This field specifies the additional points added each time decreased correction is invoked. 0 - 15
	11:8	**Shrink Param - Shrink Init** Format: U4 This field specifies the initial points required to trip decreased control. 0 - 15
	7:4	**Grow Param - Grow Resistance** Format: U4 This field specifies the additional points added each time increased correction is invoked. 0 - 15
	3:0	**Grow Param - Grow Init** Format: U4 This field specifies the initial points required to trip increased control. 0 - 15
9 Encoder Only	31	**RoundInterEnable** Format: Enable When this bit is not set, RoundInter defaults to 2.
	30:28	**RoundInter** Format: U3 Rounding precision for Inter quantized coefficients 000b: +1/16 **[Default]** 001b: +2/16 010b: +3/16 011b: +4/16 100b: +5/16 101b: +6/16 110b: +7/16

111b	+8/16

27 RoundIntraEnable

Format:	Enable

When this bit is not set, RoundIntra defaults to 4.

26:24 RoundIntra

Format:	U3

Rounding precision for Intra quantized coefficients

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

Format:	U4

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

upper_midpt.

Value	Name
0 - 15	

7:4 Correct 2

Format:	U4

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

Value	Name
0 - 15	

3:0 Correct 1

Format:	U4

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

Value	Name
0 - 15	

10 Encoder Only

31:28 ClampValues - CV7

27:24 CV6

23:20 CV5

19:16 CV4

15:12 CV3

11:8 CV2

7:4 CV1

3:0 CV0 - Clamp Value 0

Format:	U4

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

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

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

none	none	CV7	CV6	CV5	CV4	CV3	CV3
none	CV7	CV6	CV5	CV4	CV3	CV3	CV2

CV7	CV6	CV5	CV4	CV3	CV3	CV2	CV2
CV6	CV5	CV4	CV3	CV3	CV2	CV2	CV1
CV5	CV4	CV3	CV3	CV2	CV2	CV1	CV1
CV4	CV3	CV3	CV2	CV2	CV1	CV1	CV0
CV3	CV3	CV2	CV2	CV1	CV1	CV0	CV0
CV3	CV2	CV2	CV1	CV1	CV0	CV0	CV0

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

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

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

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

Value	Name
0 - 15	

MFX_AVC_WEIGHTOFFSET_STATE

MFX_AVC_WEIGHTOFFSET_STATE

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 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**

MFX_AVC_WEIGHTOFFSET_STATE

		Format: MBZ
	0	**Weight and Offset Select** It must be set in consistent with the WeightedPredFlag and WeightedBiPredIdc in the Img_State command. This parameter is specified for Intel interface only. For implicit even though only one entry may be used, still loading the whole 32-entry table.

Value	Name	Description
0	Weight and Offset L0 table	The list that followed is associated with the weight and offset for RefPicList L0
1	Weight and Offset L1 table	The list that followed is associated with the weight and offset for RefPicList L1

2..97	3071:0	**WeightOffset** WeightOffset[L=L0=0 or L1=1][i=0 to 31][Y=0/Cb=1/Cr=2][weight=0/offset=1] WeightOffset[L][i=0][Y=0][Weight=0], WeightOffset[L][i=0][Y=0][Offset=1] WeightOffset[L][i=0][Cb=1][Weight=0], WeightOffset[L][i=0][Cb=1][Offset=1] WeightOffset[L][i=0][Cr=2][Weight=0], WeightOffset[L][i=0][Cr=2][Offset=1]: WeightOffset[L][i=31][Y=0][Weight=0], WeightOffset[L][i=31][Y=0][Offset=1] WeightOffset[L][i=31][Cb=1][Weight=0], WeightOffset[L][i=31][Cb=1][Offset=1] WeightOffset[L][i=31][Cr=2][Weight=0], WeightOffset[L][i=31][Cr=2][Offset=1] Format for explicit: Both Weight and Offset are S15 in two's compliment, with a valid range from -128 to 128 Format for implicit: S15 This set of fields is always present whenever this command is issued. The full table, one entry for each reference picture, is always specified. Any reference list L0/L1[i] that does not exist, the corresponding weight and offset are set to 0. Weight and Offset are 2 byte each. Apair of Weight and Offset forms a dword, with Weight in the LOWER word and Offset in the HIGHER word. WeightOffset[L0=0][i=0 to 31][Y=0] (i.e. luma_weight_l0[i]) are specified for the weighting and offset factors applied to the luma prediction value for list 0 prediction using RefPicList0[i] (one-to-one correspondence in i). When luma_weight_l0_flag (Slice Header syntax element) is equal to 1, the value of luma_weight_l0[i] shall be in the range of -128 to 127. When luma_weight_l0_flag is equal to 0, luma_weight_l0[i] shall be inferred to be equal to 2luma_log2_weight_denom for RefPicList0[i]. luma_log2_weight_denom is a Slice Header syntax element. WeightOffset[L0=0][i=0 to 31][Cb=1] (i.e. chromaCb_weight_l0[i]) are specified for the weighting and offset factors applied to the chroma Cb prediction values for list 0 prediction using RefPicList0[i] (one-to-one correspondence in i). When chroma_weight_l0_flag (Slice Header syntax element) is equal to 1, the value of chromaCb_weight_l0[i] shall be in the range of -128 to 127. When chroma_weight_l0_flag is equal to 0, chromaCb_weight_l0[i] shall be inferred to be equal to 2chroma_log2_weight_denom for RefPicList0[i]. chroma_log2_weight_denom is a Slice Header syntax element. WeightOffset[L0=0][i=0 to 31][Cr=2] (i.e. chromaCr_weight_l0[i]) are specified for the weighting and offset factors applied to the chroma Cr prediction values for list 0 prediction using RefPicList0[i] (one-to-one correspondence in i). When chroma_weight_l0_flag (Slice Header syntax element) is equal to 1, 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

MFX_AVC_WEIGHTOFFSET_STATE		
		2chroma_log2_weight_denom for RefPicList0[i].

MFX_BSP_BUF_BASE_ADDR_STATE

MFX_BSP_BUF_BASE_ADDR_STATE

Source: VideoCS

Length Bias: 2

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Pipeline Format: OpCode
	26:24	**Media Command Opcode** Default Value: 0h MFX_COMMON_STATE Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 4h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 8h Excludes DWord (0,1)

MFX_BSP_BUF_BASE_ADDR_STATE

		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. This Row Store buffer must be 64-byte cacheline aligned. Hardware uses the horizontal address of the current macroblock to address this Row Store. For AVC BSD, 2 cacheline (CL) per MB when in MBAFF mode (row of MB pair); 1 CL per MB for non-MBAFF. So, to support 256 MBs per row (4K screen resolution), 2 * 256 * 64 bytes = 32,768 bytes are required. Cacheline alignment should be followed. For AVC MPC, 1 cachline for non-MBAFF, 2 cachelines for MBAFF per MB. For VC1, the BSD row store is 512-bit (one cacheline) per MB, times the number of MBs per picture MB row. **Programming Notes** This is one of the four RowStore Scratch Buffers which can programmed to use the internal Media Storage (total size 640 CacheLine). When Deblocking Filter Row Store Scratch Buffer Cache Select is programmed to "1", this will be stored inside MFX Media Internal Storage. Driver then needs to program this Base Address between 0 to 639, indicating starting cacheline address location for this buffer. Driver needs to make sure the whole buffer fits into Media Internal Storage. *(Notes: 1 cachelines per MB for non-mbaff; 2 cachelines per MB pair for mbaff, and the buffer needs to have enough space for 1 MB (pair) row).*
	5:0	**Reserved** Format: MBZ
2	31:16	**Reserved** Format: MBZ Reserved for 64-bit address extension.
	15:0	**BSD/MPC Row Store Scratch Buffer Base Address - Read/Write [47:32]** **Description** This field is for the upper range of BSD/MPC Row Store Scratch Buffer Base Address. This field is used for 48-bit addressing.
3	31:15	**Reserved**

	Format:	MBZ

14:13 BSD/MPC Row Store Scratch Buffer - Tiled Resource Mode

Format:	U2

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

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

12 BSD/MPC Row Store Scratch Buffer Cache Select

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

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

11 Reserved

Format:	MBZ

10:9 Reserved

Format:	MBZ

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

Format:	U2

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

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

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

Format:	U6

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

		populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.
	0	**Reserved** | | |
4	31:6	**MPR Row Store Scratch Buffer Base Address - Read/Write (Decoder Only)** This field provides the base address of the scratch buffer used by decoder's MPR unit to store MB information of the previous row for decoding each macroblock in the current row. It is a private buffer used by the MPR hardware only. Its content is not accessible by software. **Programming Notes** The MPR Row Store buffer must be 64-byte cacheline aligned.Hardware uses the horizontal address of each macroblock to address the MPR Row Store. Except ILDB Control Data, all other operations does not cross slice boundary. This field is specified in frame-level.2 cacheline (CL) per MB when in MBAFF mode (row of MB pair); 1 CL per MB for non-MBAFF, So, to support 256 MBs per row (4K screen resolution), 2 * 256 * 64 bytes = 32,768 bytes are required. Cacheline alignment should be followed. This field is only valid for AVC decoder mode This is one of the four RowStore Scratch Buffers which can programmed to use the internal Media Storage (total size 640 CacheLine). When Deblocking Filter Row Store Scratch Buffer Cache Select is programmed to "1", this will be cache inside MFX Media Internal Storage. Driver then needs to program this Base Address between 0 to 639, indicating starting cachelines address location for this buffer. Driver needs to make sure the whole buffer fits into Media Internal Storage *(Notes: 1 cachelines per MB for non-mbaff; 2 cachelines per MB pair for mbaff, and the buffer needs to have enough space for 1 MB (pair) row).*
	5:0	**Reserved** | | | Format: MBZ
5	31:16	**Reserved** | | | Format: MBZ Reserved for 64-bit address extension.
	15:0	**MPR Row Store Scratch Buffer Base Address - Read/Write [47:32]** | | | This field is for the upper range of MPR Row Store Scratch Buffer Base Address. This field is used for 48-bit addressing.
6	31:15	**Reserved** | | | Format: MBZ

14:13 **MPR Row Store Scratch Buffer - Tiled Resource Mode**

Format:	U2

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

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

12 **MPR Row Store Scratch Buffer Cache Select**

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

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

11 **Reserved**

Format:	MBZ

10:9 **Reserved**

Format:	MBZ

8:7 **MPR Row Store Scratch Buffer - Arbitration Priority Control**

Format:	U2

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

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

6:1 **MPR Row Store Scratch Buffer - Index to Memory Object Control State (MOCS) Tables**

Format:	U6

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

MFX_BSP_BUF_BASE_ADDR_STATE

	0	**Reserved**
7	31:6	**Bitplane Read Buffer Base Address** It must be cacheline aligned (i.e. 64 bytes address boundary), so lower bit 0 to 5 are used for controlling information. Bitplane buffer is a linear buffer. In VC1 Long format, it is written by an application. In VC1 Short Format, it is written and read by H/W only.For VC1 intel Long Format : it is a read-only bufferFor VC1 DXVA2 Short Format : it is a write and a read buffer. This field is only valid for VC1 decoder mode.
	5:0	**Reserved** Format: MBZ
8	31:16	**Reserved** Format: MBZ Reserved for 64-bit address extension.
	15:0	**Bitplane Read Buffer Base Address - Read/Write [47:32]** This field is for the upper range of Bitplane Read Buffer Base Address. This field is used for 48-bit addressing.
9	31:15	**Reserved** Format: MBZ
	14:13	**Bitplane Read Buffer - Tiled Resource Mode** Format: U2 **For Media Surfaces:** This field specifies the tiled resource mode.

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

	12:11	**Reserved**

MFX_BSP_BUF_BASE_ADDR_STATE

	Format:	MBZ

10:9 Reserved

Format:	MBZ

8:7 Bitplane Read Buffer - Arbitration Priority Control

Format:	U2

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

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

6:1 Bitplane Read Buffer - Index to Memory Object Control State (MOCS) Tables

Format:	U6

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

0 Reserved

MFX_DBK_OBJECT

MFX_DBK_OBJECT

Source: VideoCS

Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_DBK_OBJECT Format: OpCode
	26:24	**Media Command Opcode** Default Value: 0h Common Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 9h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 0Bh Excludes DWord (0,1) Format: =n Note: Regardless of the mode, inline data must be present in this command
1	31:6	**Pre Deblocking Source Address** Format: GraphicsAddress[31:6] Specifies the 4K byte aligned frame buffer address for outputting the non-filtered reconstructed YUV picture (i.e. output of final adder in each codec standard, and prior to the deblocking filter unit).
	5:0	**Reserved**

		Format: MBZ
2	31:16	**Reserved** Format: MBZ
	15:0	**Pre Deblocking Source Address High** Format: GraphicsAddress[47:32] This field is for the upper range of Pre-Deblocking Source Address. This field is used for 48-bit addressing.
3	31:15	**Reserved** Format: MBZ
	14:13	**Pre Deblocking Source - Tiled Resource Mode** **For Media Surfaces:** This field specifies the tiled resource mode.

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

	12:11	**Reserved** Format: MBZ
	10	**Pre Deblocking Source - Memory Compression Mode** Distinguishes Vertical from Horizontal compression. Please refer to vol1a **Memory Data Formats** chapter, **Media Memory Compression** section for more details.

Value	Name
1	Vertical Compression Mode

	9	**Pre Deblocking Source - Memory Compression Enable** Format: Enable Memory compression will be attempted for this surface.

Value	Name
0	Compression Disable

		1 Compression Enable
	8:7	**Pre Deblocking Source - Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. Value / Name: 00b: Highest priority 01b: Second highest priority 10b: Third highest priority 11b: Lowest priority
	6:1	**Pre Deblocking Source - Index to Memory Object Control State (MOCS) Tables** The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.
	0	**Reserved**
4	31:6	**Deblocking Control Address** Format: GraphicsAddress[31:6] Specifies the 4K byte aligned frame buffer address as input MB-level deblocking parameters to control the way hardware deblock the each micro-block. One 512-bit cacheline is allocated for each Macroblock in raster scan order.
	5:0	**Reserved** Format: MBZ
5	31:16	**Reserved** Format: MBZ
	15:0	**Deblocking Control Address High** Format: GraphicsAddress[47:32] This field is for the upper range of Deblocking Control Address (DeblockCntrlAddr). This field is used for 48-bit addressing.
6	31:15	**Reserved**

	Format:	MBZ

14:13 Deblocking Control - Tiled Resource Mode

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

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

12:11 Reserved

Format:	MBZ

10 Deblocking Control - Memory Compression Mode

Distinguishes Vertical from Horizontal compression. Please refer to vol1a **Memory Data Formats** chapter, **Media Memory Compression** section for more details.

Value	Name
0	Horizontal Compression Mode
1	Vertical Compression Mode

9 Deblocking Control - Memory Compression Enable

Format:	Enable

Memory compression will be attempted for this surface.

Value	Name
0	Compression Disable

8:7 Deblocking Control - Arbitration Priority Control

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

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

6:1 Deblocking Source - Index to Memory Object Control State (MOCS) Tables

The index to define the L3 and system cache memory properties. The details of the controls are

		further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.
	0	**Reserved** | | |
7	31:6	**Deblocking Destination Address** Format: GraphicsAddress[31:6] Specifies the 4K byte aligned frame buffer address for outputting the post-loop filtered reconstructed YUV picture (i.e. output of the deblocking filter unit)
	5:0	**Reserved** | | | Format: MBZ
8	31:16	**Reserved** | | | Format: MBZ
	15:0	**Deblocking Destination Address High** | | | Format: GraphicsAddress[47:32] This field is for the upper range of Deblocking Destination Address. This field is used for 48-bit addressing.
9	31:15	**Reserved** | | | Format: MBZ
	14:13	**Deblocking Destination - Tiled Resource Mode** | | | **For Media Surfaces:** This field specifies the tiled resource mode. Value / Name / Description: 0h / TRMODE_NONE / No tiled resource 1h / TRMODE_TILEYF / 4KB tiled resources 2h / TRMODE_TILEYS / 64KB tiled resources 3h / Reserved /
	12:11	**Reserved** | | | Format: MBZ

	10	**Deblocking Destination - Memory Compression Mode** Distinguishes Vertical from Horizontal compression. Please refer to vol1a **Memory Data Formats** chapter, **Media Memory Compression** section for more details. **Value** 0 — Horizontal Compression Mode
	9	**Deblocking Destination - Memory Compression Enable** Format: Enable Memory compression will be attempted for this surface. **Value** 0 — Compression Disable
	8:7	**Deblocking Destination - Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. **Value** 00b — Highest priority 01b — Second highest priority 10b — Third highest priority 11b — Lowest priority
	6:1	**Deblocking Destination - Index to Memory Object Control State (MOCS) Tables** The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.
	0	**Reserved**
10	31:6	**Deblock Row Store Address** Format: GraphicsAddress[31:6] This field provides the base address of the scratch buffer (read and write) used by the deblocking filter unit to store MB information of the previous row for filtering of each macroblock in the current row. The Deblocking Filter Row Store buffer must be 64-byte cacheline aligned. Hardware uses the horizontal address of the current macroblock to address the Deblocking Filter Row Store.
	5:0	**Reserved**

		Format: MBZ
11	31:16	**Reserved** Format: MBZ Reserved for 64-bit address extension.
	15:0	**Deblock Row Store Address High** Format: GraphicsAddress[47:32] This field is for the upper range of Deblock Row Store Address (DeblockRowStoreAddr). This field is used for 48-bit addressing.
12	31:15	**Reserved** Format: MBZ
	14:13	**Deblock Row Store - Tiled Resource Mode** **For Media Surfaces:** This field specifies the tiled resource mode.

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

	12	**Reserved** Format: MBZ
	11	**Reserved** Format: MBZ
	10	**Deblock Row Store - Memory Compression Mode** Distinguishes Vertical from Horizontal compression. Please refer to vol1a **Memory Data Formats** chapter, **Media Memory Compression** section for more details.

Value	Name
0	Horizontal Compression Mode

	9	**Deblock Row Store - Memory Compression Enable**

	Format:	Enable

Memory compression will be attempted for this surface.

Value	Name
0	Compression Disable

8:7 Deblock Row Store - Arbitration Priority Control

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

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

6:1 CoeffProbability StreamIn Address - Index to Memory Object Control State (MOCS) Tables

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

0 Reserved

MFX_FQM_STATE

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**

MFX_FQM_STATE

		Format: MBZ
	1:0	**AVC** Exists If: //AVC- Decoder Only **For AVC QM Type**: This field specifies which Quantizer Matrix is loaded. **Value** / **Name** 0 AVC_4x4_Intra_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs) 1 AVC_4x4_Inter_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs) 2 AVC_8x8_Intra_MATRIX 3 AVC_8x8_Inter_MATRIX
	1:0	**MPEG2** Exists If: //MPEG2- Decoder Only **For MPEG2 QM Type**: This field specifies which Quantizer Matrix is loaded. **Value** / **Name** 0 MPEG_INTRA_QUANTIZER_MATRIX 1 MPEG_NON_INTRA_QUANTIZER_MATRIX 2-3 Reserved
	1:0	**JPEG** Exists If: //JPEG- Encoder Only **For JPEG QM Type**:This field specifies which Quantizer Matrix is loaded. **Value** / **Name** 0 JPEG_Luma_Y_QUANTIZER_MATRIX (or R) 1 JPEG_Chroma_Cb_QUANTIZER_MATRIX (or G) 2 JPEG_Chroma_Cr_QUANTIZER_MATRIX (or B) **Programming Notes** For JPEG encoder, each quantization element presents 16-bit 1/QM[i][j]. In RGB encoding, because the order input image components can be RGB, GBR, BGR, YUV, the value 0 is used for the first image component, the value 1 is used for the second image component, and the value 2 is used for the third image component.
2..33	1023:0	**Forward Quantizer Matrix** The format of a Quantizer Matrix is an 8x8 matrix in raster order. Each element is an unsigned byte.

MFX_IND_OBJ_BASE_ADDR_STATE

MFX_IND_OBJ_BASE_ADDR_STATE

Source: VideoCS

Length Bias: 2

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

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

While the use of base address is unconditional, the indirection can be effectively disabled by setting the base address to zero.
For decoder, there are:

- 1 read-only per-slice indirect object in the BSD_OBJECT Command, and
- 2 read-only per-MB indirect objects in the IT_OBJECT Command.

For decoder: the Video Command Streamer (VCS) will perform the memory access bound check automatically using the corresponding MFC Indirect Object Access Upper Bound specification. If any access is at or beyond the upper bound, zero value is returned. The request to memory is still being sent, but the corresponding codec's BSD unit will detect this condition and perform the zeroing return. If the Upper Bound is turned off, the beyond bound request will return whatever on the bus (invalid data).
For encoder, there are:

- 1 read-only per-MB indirect object in the PAK_OBJECT Command, and
- 1 write-only per-slice indirect object in the PAK Slice_State Command

For encoder: whenever an out of bound address accessing request is generated, VMX will detect such requests and snap the address to the corresponding [indirect object base address + indirect data start address]. VMX will return all 0s as the data to the requestor. NotationDefinitionPhysicalAddress[n:m] Corresponding bits of a physical graphics memory byte address (not mapped by a GTT) GraphicsAddress[n:m] Corresponding bits of an absolute, virtual graphics memory byte address (mapped by a GTT).

DWord	Bit	Description	
0	31:29	**Command Type**	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode

MFX_IND_OBJ_BASE_ADDR_STATE

	28:27	**Pipeline** Default Value: 2h MFX_IND_OBJ_BASE_ADDR_STATE Format: OpCode
	26:24	**Common Opcode** Default Value: 0h MFX_IND_OBJ_BASE_ADDR_STATE Format: OpCode
	23:21	**Sub OpcodeA** Default Value: 0h MFX_IND_OBJ_BASE_ADDR_STATE Format: OpCode
	20:16	**SubOpcodeB** Default Value: 3h MFX_IND_OBJ_BASE_ADDR_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 0018h Excludes DWord (0,1) Format: =n Total Length - 2
1..2	63:0	**MFX Indirect Bitstream Object - Base Address** Format: **SplitBaseAddress4KByteAligned** 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.
3	31:0	**MFX Indirect Bitstream Object - Attributes** Format: **MemoryAddressAttributes**
4..5	63:0	**MFX Indirect Bitstream Object - Upper Bound** Format: **SplitBaseAddress4KByteAligned** This field specifies the 4K-byte aligned (exclusive) maximum Graphics Memory address access by the indirect data object in the MFD_XXX_BSD_OBJECT command for the compressed Slice Data. Indirect data accessed at this address and beyond will return as 0 by the hardware. Setting this field to 0 will cause this range check to be ignored. If non-zero, this address must be greater than the MFX Indirect Bitstream ObjectBase Address state. Hardware ignores this field if indirect data is not present, i.e. the Indirect Data Length field of the MFD_XXX_BSD_OBJECT command is set to 0.This field is only valid in MPEG2, AVC, VP8, and VC1 decoder VLD mode. **Programming Notes**

		For **VP8 Encoder**, this field is corresponding to **MFC Indirect PAK-BSE Object - Access Upper Bound in DW24, DW25**. Please program Indirect bitstream upperbound in this field the same as DW24, DW25.
6..7	63:0	**MFX Indirect MV Object - Base Address** Format: **SplitBaseAddress4KByteAligned** 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
8	31:0	**MFX Indirect MV Object - Attributes** Format: **MemoryAddressAttributes**
9..10	63:0	**MFX Indirect MV Object - Upper Bound** Format: **SplitBaseAddress4KByteAligned** This field specifies the 4K-byte aligned (exclusive) maximum Graphics Memory address access by the indirect data object in the MFC_AVC_PAK_OBJECT / MFD_IT_OBJECT command for the per-MB MV data. Indirect data accessed at this address and beyond will return as 0 by the hardware. Setting this field to 0 will cause this range check to be ignored. If non-zero, this address must be greater than the MFX Indirect MV Object Base Address state. Hardware ignores this field if indirect data is not present, i.e. the Indirect Data Length field of the MFC_AVC_PAK_OBJECT / MFD_IT_OBJECT command is set to 0.This field is only valid in AVC encoder mode or in AVC decoder IT mode.
11..12	63:0	**MFD Indirect IT-COEFF Object - Base Address** Format: **SplitBaseAddress4KByteAligned** 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.
13	31:0	**MFD Indirect IT-COEFF Object - Attributes** Format: **MemoryAddressAttributes**
14..15	63:0	**MFD Indirect IT-COEFF Object - Upper Bound** Format: **SplitBaseAddress4KByteAligned**

		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.
16..17	63:0	**MFD Indirect IT-DBLK Object - Base Address** Format: **SplitBaseAddress4KByteAligned** 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.
18	31:0	**MFD Indirect IT-DBLK Object - Attributes** Format: **MemoryAddressAttributes**
19..20	63:0	**MFD Indirect IT-DBLK Object - Upper Bound** Format: **SplitBaseAddress4KByteAligned** 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.
21..22	63:0	**MFC Indirect PAK-BSE Object - Base Address** Format: **SplitBaseAddress4KByteAligned** 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.
23	31:0	**MFC Indirect PAK-BSE Object - Attributes** Format: **MemoryAddressAttributes**
24..25	63:0	**MFC Indirect PAK-BSE Object - Upper Bound**

Format:	**SplitBaseAddress4KByteAligned**

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 ignored If non-zero, this address must be greater than the MFC Indirect PAK-BSE Object Base Address state. This field is only valid in AVC encoder mode.

Programming Notes

For VP8 Encoder, this field should be programmed the same at both DW4, DW5 **MFX Indirect Bitstream Object - Access Upper Bound** as well as DW24, DW25.

MFX_JPEG_HUFF_TABLE_STATE

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.

MFX_JPEG_HUFF_TABLE_STATE

		Value / **Name** / **Description** 0 / Y / Huffman table for Y
2..4	95:0	**DC_BITS (12 8-bit array)** The number of DC Huffman codes of length i, where i is 1~12
5..7	95:0	**DC_HUFFVAL (12 8-bit array)** The value associated with each DC Huffman code of length i.
8..11	127:0	**AC_BITS (16 8-bit array)** the list of Li, number of Huffman codes of length i, where i is 1~16
12..51	1279:0	**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

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	**Reserved** Exists If: //Encoder Only Format: MBZ
	31:21	**Reserved** Exists If: //Decoder Only

	Format:	MBZ

30:26 Pixels In Horizontal Last MCU

Exists If:	//Encoder Only

The number of pixels in the last MCU in a row MCUs. This information is used for completion of partial MCU.

25:21 Pixels In Vertical Last MCU

Exists If:	//Encoder Only

The number of pixels in the last MCU in a column MCUs. This information is used for completion of partial MCU.

20 Vertical Up-Sampling Enable

Exists If:	//Decoder Only

Only applied to chroma blocks. This flag is used for 2:1 vertical up-sampling for chroma 420 and outputting chroma422 YUY2 or UYVY format. To enable this flag, the input should be interleaved Scan, **InputFormatYUV** should be set to YUV420, and **OutputFormatYUV** should be set to YUY2 or UYVY.

Value	Name	Description
0b		no up-sampling
1b		2:1 vertical up-sampling

20:12 Reserved

Exists If:	//Encoder Only
Format:	MBZ

19 Reserved

Exists If:	//Decoder Only

18 Horizontal Down-Sampling Enable

Exists If:	//Decoder Only

Only applied to chroma blocks. This flag is used for 2:1 horizontal down-sampling for chroma 422 and outputting chroma420 NV21 format. To enable this flag, the input should be interleaved Scan, **InputFormatYUV** should be set to YUV422V_2Y or YUV422V_4Y, and **OutputFormatYUV** should be set to NV12.

Value	Name	Description

0b		no down-sampling
1b		2:1 horizonatl down-sampling

17 **Vertical Down-Sampling Enable**

Exists If:	//Decoder Only

Only applied to chroma blocks. This flag is used for 2:1 vertical down-sampling for chroma 422 and outputting chroma420 NV21 format. To enable this flag, the input should be interleaved Scan, **InputFormatYUV** should be set to YUV422H_2Y or YUV422H_4Y, and **OutputFormatYUV** should be set to NV12.

Value	Name	Description
0b		no down-sampling
1b		2:1 vertical down-sampling

16 **Average Down Sampling**

Exists If:	//Decoder Only

This flag is used to select a down-sampling method when **VertDownSamplingEnb** or **HoriDownSamplingEnb** is set to 1.

Value	Name	Description
0b		Drop every other line (or column) pixels
1b		Average neighboring two pixels

15:12 **Reserved**

Exists If:	//Decoder Only
Format:	MBZ

11:8 **Input Surface Format YUV**

Exists If:	//Encoder Only

This field specifies the surface format to read a YUV image data

Value	Name	Description
0000b		Reserved
0001b	NV12	NV12 for chroma 4:2:0
0010b	UYVY	UYVY for chroma 4:2:2
0011b	YUY2	YUY2 for chroma 4:2:2
0100b	Y8	Y8 for chroma400 Y-only image
0101b	RGB	RGB or YUV for chroma 4:4:4

Programming Notes
This field should be set accordingly for **SurfaceFormat** in MFX_SURFACE_STATE command.

R8G8B8A8_UNORM in this field is used for encoding RGB and YUV chroma 4:4:4. For RGB input, any order of image components R, G, B (e.g., RGB, GBR, BGR, YUV) will be acceptable as far as the order of Quantization tables and Huffman tables match the order of image components.

11:8 **Output Format YUV**

Exists If:	//Decoder Only

This field specifies the surface format to write the decoded JPEG image. Note that any non-interleaved JPEG input should be set to "0000". For the interleaved input Scan data, it can be set either "0000" or the corresponding format.

Value	Name	Description
0000b		3 separate plane for Y, U, and V respectively
0001b		NV12 for chroma 4:2:0
0010b		UYVY for chroma 4:2:2
0011b		YUY2 for chroma 4:2:2

Programming Notes

The **MFX_SURFACE_STATE** command should be set accordingly for each **OutputFormatYUV**.
For NV12, **Surface Format** = 4 (PLANAR_420_8)
For YUY2, **Surface Format** = 0 (YCRCB_NORMAL)
For UYVY, **Surface Format** = 3 (YCRCB_SWAPY)
NV12 (0001b) can be set only when Y, U, V are interleaved in a single Scan data with the following cases

- **InputFormatYUV** is YUV420 and **VertDownSamplingEnb** is disalbed
- **InputFormatYUV**is YUV422H_2Y or YUV422H_4Y, and **VertDownSamplingEnb** is enabled

UYVY (0010b) and YUY2 (0011b) can be set only when Y, U, V are interleaved in a single Scan data with the following cases

- **InputFormatYUV** is YUV420 and **VertUpSamplingEnb** is enabled
- **InputFormatYUV** is YUV422H_2Y or YUV422H_4Y and **VertUpSamplingEnb** is disabled

7:6 **Reserved**

Exists If:	//Decoder Only
Format:	MBZ

7:6 **Reserved**

Exists If:	//Encoder Only
Format:	MBZ

5:4 **Rotation**

Exists If:	//Decoder Only

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

Programming Notes
Rotation can be set to 01b, 10b, or 11b when OutputFormatYUV is set to 0000b. For other OutputFormatYUV, Rotation is not allowed.

5:3 **Reserved**

Exists If:	//Encoder Only
Format:	MBZ

3 **Reserved**

Exists If:	//Decoder Only
Format:	MBZ

2:0 **Input Format YUV**

Exists If:	//Decoder Only
Format:	U3

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

2:0 **Output MCU Structure**

Exists If:	//Encoder Only

Output MCU Structure(**OutputMcuStructure**) should be set accordingly for each Input Surface Format YUV(**InputSurfaceFormatYUV**):

- If **InputSurfaceFormatYUV** is set to NV12, **OutputMCUStructure** is set to YUV420.
- If **InputSurfaceFormatYUV** is set to UYVY or YUY2, **OutputMCUStructure** is set to YUV422H_2Y.
- If **InputSurfaceFormatYUV** is set to Y8, **OutputMCuStructure** is set to YUV400.
- If **InputSurfaceFormatYUV** is set to RGB (or GBR, BGR, YUV), **OutputMCuStructure** is set to RGB.
- If **InputSurfaceFormatYUV** is set to RGB, the order of encoded blocks in MCU will be same as the order of input image components. If the order of input image components is RGB (or GBR, BGR, YUV), then the order of blocks will be RGB (or GBR, BGR, YUV respectively).

Value	Name	Description
0	YUV400	Grayscale Image
1	YUV420	Both horizontally and vertically chroma 2:1 subsampled
2	YUV422H_2Y	Horizontally chroma 2:1 subsampled - horizontal 2 Y-blocks, 1 U and 1 V block
3	RGB	RGB or YUV444: No subsample
4		
5		
6		
7		

2 | 31:30 **Reserved**

Exists If:	//Decoder Only
Format:	MBZ

31:29 **Reserved**

Exists If:	//Encoder Only
Format:	MBZ

29 **Output Pixel Normalize**

Exists If:	//Decoder Only

JPEG decoded output pixels for Y and U/V in order to adjust display YUV range.

Value	Name	Description	Exists If
0		No Normalization	
1		Normalize output pixels from [0,255] to [16,235]	//Y
1		Normalize output pixels from [0,255] to [16,239]	//U/V

MFX_JPEG_PIC_STATE

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.

28:16 **Frame Height In Blks Minus 1**

Exists If:	//Encoder Only
Format:	U13-1

(The number of blocks in height) - 1. This value is calculated using the number of lines Y and vertical sampling factor of the first component V1 in Frame header. See the note following this table.
For interleaved components: (((Y + (V1*8 -1)) / (V1*8)) * V1) - 1 For non-interleaved components: ((Y + 7) / 8) - 1

15:13 **Reserved**

Exists If:	//Decoder Only
Format:	MBZ

15:13 **RoundingQuant**

Exists If:	//Encoder Only

Rounding value applied to quantization output

Value	Name	Description
000b	**[Default]**	1/2
001b		(1/2 - 1/128)
010b		(1/2 + 1/128)
011b		(1/2 - 1/64)
100b		(1/2 + 1/64)
101b		(1/2 - 1/32)
110b		(1/2 - 1/16)
111b		(1/2 - 1/8)

12:0 **Frame Width In Blocks Minus 1**

Exists If:	//Decoder Only
Format:	U13-1

(The number of blocks in width) - 1. This value is calculated using the number of samples per

		line X and horizontal sampling factor of the first component H_1 in Frame header. See the note following this table. For interleaved components, (((X + (H_1 *8 -1)) / (H_1 *8)) * H_1) - 1. For non-interleaved components, ((X + 7) / 8) - 1.
	12:0	**Frame Width In Blks Minus 1** Exists If: //Encoder 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 H1 in Frame header. See the note following this table. For interleaved components: (((X + (H1 *8 -1)) / (H1 *8)) * H1) - 1 For non-interleaved components: ((X + 7) / 8) - 1

MFX_MPEG_TS_CONTROL command

MFX_MPEG_TS_CONTROL command

Source: VideoCS

Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h Command Type Format: Opcode
	28:27	**Pipeline** Default Value: 2h Format: Opcode
	26:24	**Opcode** Default Value: 0h Format: Opcode
	23:21	**SubOpA** Default Value: 2h Format: Opcode
	20:16	**SubOpB** Default Value: Bh Format: Opcode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Value: 3h — Name: DWORD_COUNT_n **[Default]** — Description: Total length - 2 (excludes DWord0 and DWord1)
1	31:30	**Reserved** Format: MBZ
	29	**payload_unit_start_indicator control** Programming Notes: This bit should be programmed zero always.
	28	**Additional Copy info flag in PES header**
	27	**DSM trick mode flag in PES header**

MFX_MPEG_TS_CONTROL command		
	26	**Original or flag in PES header**
	25	**Copy Right flag in PES header**
	24	**Output TS packet grouping select** 0: Return all packets continuously 1: Return 7 packets in 2K aligned buffer (with the remaining bits between the end of the 7th packet and the end of the 2K buffer including the rest being undefined)
	23:20	**StreamID lower Nibble** Stream ID Lower Nibble. Stream ID for Video can be 0xE0 through 0xEF. This 4 bit field indicates the last 4 bits of Stream ID in Mpeg transport stream as indicated in the DCN diagram
	19:13	**Reserved** Format: MBZ
	12:0	**Video PacketID Header Parameter** This field specifies the static program fields in MPEG header for each Video packet.
2	31:0	**PCR 90 KHz component least significant bits.**
3	31:23	**27MHz Counter** Full 8-bits of 27Mhz counter
	22:1	**Reserved** Format: MBZ
	0	**90 KHz counter MSB** Upper bit (bit 33) of 90khz counter
4	31:0	**PTS Delta** PTS Delta to be applied to 90 KHz count of PCR to generate PTS. This is a Twos complement number and added to 90 KHz PCR counter to generate PTS.
5	31:28	**Continuity Counter** This field specifies the 4b continuity counter given in the MPEGTS packet header. This should be initialized with the value that was read from MMIO at the end of the previous frame. That value will be incremented by HW so there is no need to SW to increment it. For the first frame this should be set to 0.
	27:16	**Reserved** Format: MBZ
	15:0	**MPEGTS Packet Count** This field is ignored by HW. Driver can copy MFX_PAK_MPEGTS_STATUS register from the previous frame to DW 5 of MPEG_TS_CONTROL_command using MI_STORE_REG_MEM

MFX_MPEG2_PIC_STATE

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** Format: U2 See ISO/IEC 13818-2 6.3.10 for details.
	13:12	**Picture Structure** This field specifies whether the picture is encoded in the form of a frame picture or one field (top or bottom) picture. See ISO/IEC 13818-2 6.3.10 for details.Format = MPEG_PICTURE_STRUCTURE00 = Reserved01 = MPEG_TOP_FIELD10 = MPEG_BOTTOM_FIELD11 = MPEG_FRAME
	11	**TFF (Top Field First)** When two fields are stored in a picture, this bit indicates if the top field is the first field.For a frame P picture, the value 1 indicates that the top field of the reconstructed frame is the first field output by the decoding process, the same as defined in ISO/IEC 13818-2 6.3.10. Particularly, it is used by the hardware to calculate derivative motion vectors from the dual-prime motion vectors.For a field P picture, hardware uses this bit together with the Picture Structure to determine if the current picture is the Second Field. In this case, the definition of this bit differs from ISO/IEC 13818-2 6.3.10 - software must derive the value for this bit according to the following relation:Picture Structure = top fieldPicture Structure = bottom fieldSecond Field = 0TFF = 1TFF = 0Second Field = 1TFF = 0TFF = 1
	10	**Frame Prediction Frame DCT** This field provides constraints on the DCT type and prediction type. It affects the syntax of the bitstream.
	9	**Concealment Motion Vector Flag** This field indicates if the concealment motion vectors are coded in intra macroblocks. It affects the syntax of the bitstream.
	8	**Quantizer Scale Type** Format: MPEG_Q_SCALE_TYPE This field specifies the quantizer scaling type. Value / Name / Description 0h / / MPEG_QSCALE_LINEAR 1h / / D MPEG_QSCALE_NONLINEAR esc
	7	**Intra VLC Format** This field is used by VLD
	6	**Scan Order** Format: MPEG_INVERSESCAN_TYPE

This field specifies the Inverse Scan method for the DCT-domain coefficients in the blocks of the current picture.

Value	Name	Description
0h		MPEG_ZIGZAG_SCAN
1h		MPEG_ALTERNATE_VERTICAL_SCAN

5:0 **Reserved**

2 31 **I Slice Concealment Mode**

Exists If:	//Decoder

This field controls how MPEG decoder handles MB concealment in I Slice

Value	Name	Description
0h	Intra Concealment	Using Coefficient values to handle MB concealment
1h	Inter Concealment	Using Motion Vectors to handle MB concealment

Programming Notes
If this field is set to "1", driver must provide a valid forward reference picture (both top and bottom Field must be valid)

30 **Reserved**

Format:	MBZ

29:28 **P/B Slice Concealment Mode**

Exists If:	//Decoder

This field controls how MPEG decoder handles MB concealment in P/B Slice.

Value	Name	Description
00b	INTER	If left MB is NOT Intra MB type (including skipMB), use left MB inter prediction mode [frame/field or forward/backward/bi] and MV final values as concealment. Otherwise (left MB is Intra MB), use forward reference (same polarity for field pic) with MV final values set to 0.
01b	LEFT	If left MB is NOT Intra MB type (including skipMB), use left MB inter prediction mode [frame/field or forward/backward/bi] and MV final values as concealment. Otherwise (left MB is Intra MB), use left MB dct_dc_pred[cc] values for concealment (Macroblock is concealed as INTRA MB and dct_dc_pred[cc] are DC predictor for Luma, Cr, Cb data)
10b	ZERO	Always use forward reference (same polarity for field pic) with MV final values set to 0 (Macroblock is concealed as INTER coded)
11b	INTRA	Use left MB dct_dc_pred[cc] values for concealment (Macroblock is concealed as INTRA MB and dct_dc_pred[cc] are DC predictor for Luma, Cr, Cb data

27 **Reserved**

Format:	MBZ

26:25 **P/B Slice Predicted BiDir Motion Type Override - Bi-direction MV Type Override**

Exists If:	//Decoder

This field is only applicable if the Concealment Motion Type is predicted to be Bi-directional. (It is only possible if "P/B Slice Concealment Mode" is set to "00" or "01" and left MB is a bi-directional MB).

Value	Name	Description
0h	BID	Keep Bi-direction Prediction
1h	RESERVED	
2h	FWD	Only use Forward Prediction (Backward MV is forced to invalid
3h	BWD	Only use Backward Prediction (Forward MV is forced to invalid)

24 **P/B Slice Predicted Motion Vector Override Final MV value Override**

Exists If:	//Decoder

This field is only applicable if the Concealment Motion Vectors are non-zero. It is only possible if "P/B Slice Concealment Mode" is set to "00" or "01" and left MB has non-zero motion vectors).

Value	Name	Description
0h	Predicted	Motion Vectors use predicted values
1h	ZERO	Motion Vectors force to 0

23:15 **Reserved**

Format:	MBZ

14 **LoadSlicePointerFlag - LoadBitStreamPointerPerSlice**

Exists If:	//Encoder

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

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

	13	**Reserved** Format: MBZ
	12	**Reserved** Format: MBZ
	11	**Reserved** Format: MBZ
	10:9	**Picture Coding Type** Format: MPEG_PICTURE_CODING_TYPE This field identifies whether the picture is an intra-coded picture (I), predictive-coded picture (P) or bi-directionally predictive-coded picture (B). See ISO/IEC 13818-2 6.3.9 for details. **Value** / **Name** 00b: Reserved 01b: MPEG_I_PICTURE 10b: 10 = MPEG_P_PICTURE 11b: MPEG_B_PICTURE
	8:2	**Reserved** Format: MBZ
	1	**MismatchControlDisabled** These 2 bits flag disables mismatch control of the inverse transformation for some specific cases during reference reconstruction. **Value** / **Name** / **Description** 00b: Mismatch control applies to all MBs 01b: Disable mismatch control to all intra MBs whose all AC-coefficients are zero. 10b: Disable mismatch control to all MBs whose all AC-coefficients are zero. 11b: Disable mismatch control to all MBs.
	0	**Disable Mismatch** To disable MPEG2 IDCT fixed point arithmetic correction
3	31	**Slice Concealment Disable Bit** Exists If: //Decode If VINunit detects the next slice starting position is either out-of-bound or smaller than or equal to the current slice starting position, VIN will set the current slice to be 1 MB and force VMDunit to do slice concealment on the next slice. This bit will disable this feature and the MB data from the next slice will be decoded from bitstream. **Value** / **Name** / **Description** 0h: Enable: VIN will force next slice to be concealment if detects slice

		[Default] boundary error 1h Disable VIN will not force next slice to be in concealment **Programming Notes** Driver has an option to detect the scenario given in description (above) and remove the second (out-of-order) slice. In this case, hardware will decode the first slice in completion and do concealment till the third slice. It should yield a picture with better quality this way.
	30:29	**Reserved** Format: MBZ
	28:24	**Reserved**
	23:16	**FrameHeightInMBsMinus1[7:0] (Picture Height in Macroblocks)** Format: U8
	15:8	**Reserved** Format: MBZ for future supporting width > 4K
	7:0	**FrameWidthInMBsMinus1[7:0] (Picture Width in Macroblocks)** Format: U8
4	31:16	**MinFrameWSize** Format: U16 - Minimum Frame Size [15:0] (16-bit) (Encoder Only)Minimum Frame Size is specified to compensate for intel Rate Control Currently zero fill (no need to perform emulation byte insertion) is done only to the end of the CABAC_ZERO_WORD insertion (if any) at the last slice of a picture. Intel encoder parameter. The caller should always make sure that the value, represented by Minimum Frame Size, is always less than maximum frame size FrameBitRateMax (DWORD 10 bits 29:16). This field is reserved in Decode mode. Value / Name / Description [0,0003FFFFh] / / The programmable range when MinFrameWSizeUnits is 00. [0,000FFFFFh] / / The Programmable range when MinFrameWSizeUnits is 01. [0,03FFFFFFh] / / The Programmable range when MinFrameWSizeUnits is 10. [0, FFFFFFFFh] / / The Programmable range when MinFrameWSizeUnits is 11. 0h / [Default] /
	15	**Reserved** Format: MBZ
	14:12	**RoundInterAC,** rounding precision for non-Intra AC000: +1/16001: +2/16010: +3/16011: +4/16100:

		+5/16101: +6/16110: +7/16111: +8/16
	11	**Reserved** Format: MBZ
	10:8	**RoundIntraAC** Format: U3 rounding precision for Intra AC000: +1/16001: +2/16010: +3/16011: +4/16100: +5/16101: +6/16110: +7/16111: +8/16
	7	**Reserved** Format: MBZ
	6:4	**RoundInterDC** rounding Precision for non-Intra-DC000: +1/16001: +2/16010: +3/16011: +4/16100: +5/16101: +6/16110: +7/16111: +8/16
	3	**Reserved** Format: MBZ
	2:1	**RoundIntraDC** rounding Precision for Intra-DC00: +1/801: +2/810: +3/811: +4/8
	0	**Reserved**
5	31:17	**Reserved** (for future Mask bits)
	16	**FrameSizeControlMask** Frame size conformance mask. This field is used when MacroblockStatEnable is set to 1. 0h — Do not change Slice Quantization Parameter values in MFC_MPEG2_SLICEGROUP_STATE with suggested slice QP value for frame level Rate control 1h — Replace Slice Quantization Parameter values in MFC_MPEG2_SLICEGROUP_STATE with suggested slice QP value for frame level Rate control values in MFC_IMAGE_STATUS control register.
	15:13	**Reserved**
	12	**InterMBForceCBPZeroControlMask** Format: U1 Inter MB Force CBP ZERO mask. [0, FFFFFFFFh] —
		0h — No effect

1h		Zero out all A/C coefficients for the inter MB violating Inter Confirmance

11:10 MinFrameWSizeUnits

This field is the Minimum Frame Size Units

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

9 MBRateControlMask

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

Value	Name	Description
0h		Do not change QP values of inter macroblock with suggested QP values in Macroblock Status Buffer
1h		Apply RC QP delta for all macroblock

8 Reserved

7 Reserved

Format:	MBZ

6:4 Reserved

3 FrameBitRateMinReportMask

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

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

2 FrameBitRateMaxReportMask

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

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

1 InterMBMaxSizeReportMask

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

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

IntraMBMaxSizeReportMask

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

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

Reserved

Format:	MBZ

InterMBMaxSize

Default Value:	FFFh

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

Reserved

Format:	MBZ

IntraMBMaxSize

Default Value:	FFFh

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

Reserved

Format:	MBZ

SliceDeltaQPMax[3]

Format:	S7

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

		Range: [-30,30] Value / Name: 0h — Disable 1h — Enable
	23:16	**SliceDeltaQPMax[2]** Format: S7 Range: [-30,30] This field is the Slice level delta QP for bit-count above FrameBitRateMax - above 1/8 and below 1/ 4 This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between 1/8 and ¼ of FrameBitRateMaxDelta above FrameBitRateMax, i.e., in the range of ((FrameBitRateMax+ FrameBitRateMaxDelta»3), (FrameBitRateMax+ FrameBitRateMaxDelta»2).
	15:8	**SliceDeltaQPMax[1]** Format: S7 Range: [-30,30] This field is the Slice level delta QP for bit-count above FrameBitRateMax - above1/ 4 and below 1/2 This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between ¼ and ½ of FrameBitRateMaxDelta above FrameBitRateMax, i.e., in the range of ((FrameBitRateMax+ FrameBitRateMaxDelta»2), (FrameBitRateMax+ FrameBitRateMaxDelta»1).
	7:0	**SliceDeltaQPMax[0]** Format: S7 Range: [-30,30] This field is the Slice level delta QP for bit-count above FrameBitRateMax - above 1/ 2This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is above FrameBitRateMax by more than half the distance of FrameBitRateMaxDelta , i.e., in the range of ((FrameBitRateMax+ FrameBitRateMaxDelta»1), infinite).
9 [ExistsIf]Encode Only	31:24	**SliceDeltaQPMin[3]** Format: S7 Range: [-30,30] This field is the Slice level delta QP for total bit-count below FrameBitRateMin - first 1/8 regionThis field is used to calculate the suggested slice QP into the

		MFC_IMAGE_STATUS control register when total bit count for the entire frame is less than FrameBitRateMin and greater than or equal to 1/8 the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of [(FrameBitRateMin- FrameBitRateMinDelta»3), FrameBitRateMin).
	23:16	**SliceDeltaQPMin[2]** Format: S7 Range: [-30,30] This field is the Slice level delta QP for bit-count below FrameBitRateMin - below 1/ 8 and above 1/ 4This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between one-eighth and quarter the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of [(FrameBitRateMin- FrameBitRateMinDelta»2), (FrameBitRateMin- FrameBitRateMinDelta»3)).
	15:8	**SliceDeltaQPMin[1]** Format: S7 Range: [-30,30] This field is the Slice level delta QP for bit-count below FrameBitRateMin- below 1/4 and above 1/ 2This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between quarter and half the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of [(FrameBitRateMin- FrameBitRateMinDelta»1), (FrameBitRateMin- FrameBitRateMinDelta»2)).
	7:0	**SliceDeltaQPMin[0]** Format: S7 Range: [-30,30] This field is the Slice Level Delta QP for bit-count below FrameBitRateMin - below 1/ 2This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is below FrameBitRateMin by more than half the distance of FrameBitRateMinDelta , i.e., in the range of [0, (FrameBitRateMin- FrameBitRateMinDelta»1).
10 [ExistsIf]Encode Only	31	**FrameBitrateMaxUnit** This field is the Frame Bitrate Maximum Limit Units. **Value** / **Name** / **Description** 0h / Byte / FrameBitRateMax is in units of 32 Bytes when FrameBitrateMaxUnitMode is 1 and in units of 128 Bytes if FrameBitrateMaxUnitMode is 0 1h / Kilobyte / FrameBitRateMax is in units of 4KBytes Bytes when FrameBitrateMaxUnitMode is 1 and in units of 16KBytes if

		FrameBitrateMaxUnitMode is 0
	30	**FrameBitrateMaxUnitMode** BitFiel This field is the Frame Bitrate Maximum Limit Units.dDesc **Value** / **Name** / **Description** 0h / Compatibility mode / FrameBitRateMaxUnit is in old mode (128b/16Kb) 1h / New mode / FrameBitRateMaxUnit is in new mode (32byte/4Kb)
	29:16	**FrameBitRateMax** This field is the Frame Bitrate Maximum Limit. This field along with FrameBitrateMaxUnit determines maximum allowed bits in a frame before multi-pass gets triggered (when enabled). In other words, multi-pass is triggered when the actual frame byte count exceeds this value. When FrameBitrateMaxUnitMode is 0(compatibility mode) bits 16:27 should be used, bits 28 and 29 should be 0. **Value** / **Name** / **Description** 0-512KB / / The programmable range 0-512KB when FrameBitrateMaxUnit is 0. 0-8190KB / / The programmable range 0-8190KB when FrameBitrateMaxUnit is 1.
	15	**FrameBitrateMinUnit** This field is the Frame Bitrate Minimum Limit Units. **Value** / **Name** / **Description** 0h / Byte / FrameBitRateMax is in units of 32 Bytes when FrameBitrateMinUnitMode is 1 and in units of 128 Bytes if FrameBitrateMinUnitMode is 0 1h / KiloByte / FrameBitRateMax is in units of 4KBytes Bytes when FrameBitrateMaxUnitMode is 1 and in units of 16KBytes if FrameBitrateMaxUnitMode is 0
	14	**FrameBitrateMinUnitMode** This field is the Frame Bitrate Minimum Limit Units.ValueNameDescriptionProject **Value** / **Name** / **Description** 0h / compatibility mode / FrameBitRateMaxUnit is in old mode (128b/16Kb) 1h / New Mode / FrameBitRateMaxUnit is in new mode (32byte/4Kb)
	13:0	**FrameBitRateMin** This field is the Frame Bitrate Minimum Limit ()This field along with FrameBitrateMinUnit determines minimum allowed bits in a Frame before Multi-Pass gets triggered (when enabled). In other words, multi-pass is triggered when the actual frame byte count is less than this value. When FrameBitrateMinUnitMode is 0 (compatibility mode) bits 0:11 should be used, bits 12 and 13 should be 0. Range: The programmable range 0-512KB When FrameBitrateMinUnit is in 0. Programmable range is 0-8190 KB when FrameBitrateMinUnit is in 1
11 [ExistsIf]Encode	31	**Reserved** Format: / MBZ

Only	30:16	**FrameBitRateMaxDelta** Default Value: 0h Access: None Format: U15 This field is used to select the slice delta QP when FrameBitRateMax Is exceeded. It shares the same FrameBitrateMaxUnit. The programmable range is either 0- 512KB or 4MBB in FrameBitrateMaxUnit of 128 Bytes or 16KB respectively. This field is used to select the slice delta QP when FrameBitRateMax Is exceeded. It shares the same FrameBitrateMaxUnit. When FrameBitrateMaxUnitMode is 0(compatibility mode) bits 16:27 should be used, bits 28, 29 and 30 should be 0.
	15	**Reserved** Format: MBZ
	14:0	**FrameBitRateMinDelta** This field is used to select the slice delta QP when FrameBitRateMin Is exceeded. It shares the same FrameBitrateMinUnit. When FrameBitrateMinUnitMode is 0(compatibility mode) bits 0:11 should be used, bits 12, 13 and 14 should be 0.Note: HW requires the following condition FrameBitRateMinDelta <= 2*FrameBitRateMinMust be true, otherwise it may cause unpredicted behavior. Value / Name / Description 0-1024KB / / The programmable range 0-1024KB When FrameBitrateMinUnit is in 32Bytes. 0-16380KB / / Programmable range is 0-16380KB when FrameBitrateMinUnit is in 4Kbytes.
12	31:0	**Reserved** Format: MBZ

MFX_PAK_INSERT_OBJECT

MFX_PAK_INSERT_OBJECT	
Source:	VideoCS
Length Bias:	2

Description
The MFX_PAK_INSERT_OBJECT command is the first primitive command for the AVC, MPEG2, JPEG, and VP8 Encoding Pipeline.

MFX_PAK_INSERT_OBJECT

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 location to perform the actual insertion by transferring the command inline data to the output buffer max, 32 bits at a time.
It is a variable length command as the data to be inserted are presented as inline data of this command. It is a multiple of 32-bit (1 DW), as the data bus to the bitstream buffer is 32-bit wide.
Multiple insertion commands can be issued back to back in a series. It is host software's responsibility to make sure their corresponding data will properly stitch together to form a valid H.264 bitstream.
Internally, MFX hardware will keep track of the very last two bytes' (the very last byte can be a partial byte) values of the previous insertion. It is required that the next Insertion Object Command or the next PAK Object Command to perform the start code emulation sequence check and prevention 0x03 byte insertion with this end condition of the previous insertion.
Hardware will keep track of an output bitstream buffer current byte position and the associated next bit insertion position index. Data to be inserted can be a valid H.264 NAL units or a partial NAL unit. Certain NAL unit has a minimum byte size requirement. As such the hardware will optionally (enabled by STATE Command) determines the number of CABAC_ZERO_WORD to be inserted to the end of the current NAL, based on the minimum byte size of a NAL and the actual bin count of the encoded Slice. Since prior to the CABAC_ZERO_WORD insertion, the RBSP or EBSP is already byte-aligned, so each CABAC_ZERO_WORD insertion is actually a 3-byte sequence 0x00 00 03. The inline data may have already been processed for start code emulation byte insertion, except the possibility of the last 2 bytes plus the very last partial byte (if any). Hence, when hardware performing the concatenation of multiple consecutive insertion commands, or concatenation of an insertion command and a PAK object command, it must check and perform the necessary start code emulation byte insert at the junction.The inline data is required to be byte aligned on the left (first transmitted bit order) and may or may not be byte aligned on the right (last transmitted bits).
The command will specify the bit offset of the last valid DW.Each insertion state command defines a chunk of bits (compressed data) to be inserted at a specific location of the output compressed bitstream in the output buffer.Depend on CABAC or CAVLC encoding mode (from Slice State), PAK Object Command is always ended in byte aligned output bitstream except for CABAC header insertion which is bit aligned. In the aligned cases, PAK will perform 0 filling in CAVLC mode, and 1 filling in CABAC mode.
Insertion data can include:any encoded syntax elements bit data before the encoded Slice Data (PAK Object Command) of the current SliceSPS NALPPS NALSEI NALOther Non-Slice NALLeading_Zero_8_bits (as many bytes as there is)Start Code PrefixNAL Header ByteSlice HeaderAny encoded syntax elements bit data after the encoded Slice Data (PAK Object Command) of the current Slice and prior to the next encoded Slice Data of the next Slice or prior to the end of the bistream, whichever comes firstCabac_Zero_Word or Trailing_Zero_8bits (as many bytes as there is).
Anything listed above before a Slice DataContext switch interrupt is not supported by this command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_PAK_INSERT_OBJECT Format: OpCode
	26:24	**Media Command Opcode**

MFX_PAK_INSERT_OBJECT

		Default Value: 0h MFX_COMMON Format: OpCode
	23:21	**SubOpcode A** Default Value: 2h Format: OpCode
	20:16	**SubOpcode B** Default Value: 8h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) = Variable Length in DW Format: =n Total Length - 2
1	31:18	**Reserved** Format: MBZ
	17:16	**DataByteOffset - SrcDataStartingByteOffset[1:0]** Source Data Starting Byte Position within the very first inline DW. **Programming Notes** Must be set to 0 for JPEG encoder
	15	**HeaderLengthExcludeFrmSize** In case this flag is on, bits are NOT accumulated during current access unit coding neither for Cabac Zero Word insertion bits counting or for output in MMIO register MFC_BITSTREAM_BYTECOUNT_FRAME_NO_HEADER. When using HeaderLenghtExcludeFrmSize for header insertion, the software needs to make sure that data comes already with inserted start code emulation bytes. SW shouldn't set EmulationFlag bit (Bit 3 of DWORD1 of MFX_PAK_INSERT_OBJECT).

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

Programming Notes
Must be set to 0 for JPEG encoder

	14	**Slice Header Indicator** This bit indicates if the insert object is a slice header. In the VDEnc mode, PAK only gets this command at the beginning of the frame for slice position X=0, Y=0. It internally generates the header that needs to be inserted per slice. For VDEnc mode, this bit should always be set.

Value	Name	Description

1	SLICE_HEADER	Insertion Object is a Slice Header. The command is stored internally by HW and is used for inserting slice headers.
0	LEGACY	Legacy Insertion Object command. The PAK Insertion Object command is not stored in HW.

Programming Notes
In VDENC mode, we support only Slice layer without partitioning RBSP syntax. The payload for PAK_INS_OBJ should contain only start code for Slice header followed by NAL_type and slice header (slice_header() in AVC spec). The payload for PAK_INS_OBJ shouldn't contain CABAC Byte alignment bits. HW adds these alignment bits which are part of slice_data. Example PAK_INS_OBJ payload : 00 00 01 <NAL_type> <slice_header_Byte0> .....<slice_header_Byte LAST> Any zero_bytes that are added before slice header can be inserted by any preceding general PAK_INS_OBJ.

13:8 **DataBitsInLastDW - SrCDataEndingBitInclusion[5:0]**

Source Data to be included in the very last inline DW. Follows the MSBit is the upper bit of each byte within the DW. The lower byte is actually processed first. For example, SrCDataEndingBitInclusion = 9, bit 7:0 and bit 15 are included as valid header data.

Value	Name
[1,32]	

7:4 **SkipEmulByteCnt - Skip Emulation Byte Count**

Skip emulation check for number of starting bytesIt can be programmed from 0 to 15 bytes.For example, to skip the start code that has already prefixed in the bitstream.

Programming Notes
Must be set to 0 for JPEG encoder

3 **EmulationFlag - EmulationByteBitsInsertEnable**

Value	Name	Description
0	NONE	No emulation
1	EMULATE	Instruct the hardware to perform Start Code Prefix (0x 00 00 01/02/03/00) Search and Prevention Byte (0x 03) insertion on the insertion data of this command. It is required that hardware will handle a start code prefix crossing the boundary between insertion commands, or an insertion command followed by a PAK Object command.

Programming Notes
Must be set to 0 for JPEG encoder

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

<table>
<tr><th colspan="3">MFX_PAK_INSERT_OBJECT</th></tr>
<tr><td></td><td>1</td><td>EndOfSliceFlag - LastDstDataInsertCommandFlag
No more insertion command and no more PAK-OBJECT command follows. Flush data out to memory</td></tr>
<tr><td></td><td>0</td><td>BitstreamStartReset - ResetBitStreamStartingPos
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>1</td><td>RESET</td><td>Reset the bitstream buffer insertion position to the bitstream buffer starting position.</td></tr>
<tr><td>0</td><td>INSERT</td><td>Insert the current command inline data starting at the current bitstream buffer insertion position</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>Must be set to 1 for JPEG encoder</td></tr>
</table></td></tr>
<tr><td>2..n</td><td>31:0</td><td>Insert Data PayLoad
Actual Data to be inserted to the output bitstream buffer.</td></tr>
</table>

MFX_PIPE_BUF_ADDR_STATE

MFX_PIPE_BUF_ADDR_STATE

Source: VideoCS

Length Bias: 2

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

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

MFX_PIPE_BUF_ADDR_STATE

1 — 31:6 — Pre Deblocking Destination Address

Format:	GraphicsAddress[31:6]

Specifies the 4K byte aligned frame buffer address for outputting the non-filtered reconstructed YUV picture (i.e. output of final adder in each codec standard, and prior to the deblocking filter unit). This field is ignored if PreDeblockOutEnable is set to 0 (disable).

1 — 5:0 — Reserved

2 — 31:16 — Reserved

Format:	MBZ

2 — 15:0 — Pre Deblocking Destination Address High

Format:	GraphicsAddress[47:32]

This field is for the upper range of Pre-Deblocking Destination Address. This field is ignored if **PreDeblockOutEnable** is set to 0 (disable).

3 — 31:15 — Reserved

Format:	MBZ

3 — 14:13 — Pre Deblocking - Tiled Resource Mode

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

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

3 — 12:11 — Reserved

Format:	MBZ

3 — 10 — Pre Deblocking - Memory Compression Mode

Description
Distinguishes Vertical from Horizontal compression. Please refer to vol1a**Memory Data Formats chapter -section**Media Memory Compression **for more details.**

Value	Name
0	Horizontal Compression Mode
1	Vertical Compression Mode

MFX_PIPE_BUF_ADDR_STATE

Programming Notes

This bit is not used unless Memory Compression Enable is set to "1"

All Codec except JPEG Vertical Compression Only

Video Mode	Compression Mode
AVC Frame Only (No MBAFF or Field)	Vertical
VP8 (Only Frame is supported	Vertical
VC1 (No overlap smoothing, No field)	Vertical
MPGE2 (No field)	Vertical

JPEG Decode Table

Chroma Format	Output Format	Compression Mode
422H_2Y, 422H_4Y	YUY2	Horizontal
422H_2Y, 422H_4Y	UYVY	Horizontal
422H_2Y, 422H_4Y	NV12	Not Supported
420	NV12	Vertical

9 **Pre Deblocking - Memory Compression Enable**

Format:	Enable

Memory compression will be attempted for this surface.

Value	Name
0	Compression Disable
1	Compression Enable

Programming Notes

Video Mode	Compression Enable
AVC Frame Only (No MBAFF or Field)	Yes
VP8 (Only Frame is supported	Yes
VC1 (No overlap smoothing, No field)	Yes
MPGE2 (No field)	Yes

JPEG Decode

Chroma Format	Output Format	Compression Enable
422H_2Y,422H_4Y	YUY2	Yes
422H_2Y,422H_4Y	YUY2	Yes
422H_2Y,422H_4Y	UYVY	Yes
422H_2Y, 422H_4Y, 422V_2Y, 422V_4Y	NV12	No
420	YUY2, UYVY	No
420	NV12	Yes

MFX_PIPE_BUF_ADDR_STATE

	8:7	**Pre Deblocking - Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. **Value** / **Name** 00b – Highest priority 01b – Second highest priority 10b – Third highest priority 11b – Lowest priority
	6:0	**Pre Deblocking - Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).
4	31:6	**Post Deblocking Destination Address** Format: GraphicsAddress[31:6] Specifies the 4K byte aligned frame buffer address for outputting the post-loop filtered reconstructed YUV picture (i.e. output of the deblocking filter unit)This field is ignored if PostDeblockOutEnable is set to 0 (disable).
	5:0	**Reserved**
5	31:16	**Reserved** Format: MBZ
	15:0	**Post Deblocking Destination Address High** Format: GraphicsAddress[47:32] This field is for the upper range of Post-Deblocking Destination Address. This field is ignored if **PostDeblockOutEnable** is set to 0 (disable).
6	31:15	**Reserved** Format: MBZ
	14:13	**Post Deblocking - Tiled Resource Mode** **For Media Surfaces:** This field specifies the tiled resource mode. **Value** / **Name** / **Description** 0h – TRMODE_NONE – No tiled resource 1h – TRMODE_TILEYF – 4KB tiled resources

2h	TRMODE_TILEYS	64KB tiled resources
3h	Reserved	

12:11 **Reserved**

Format:	MBZ

10 **Post Deblocking - Memory Compression Mode**

Description
Distinguishes Vertical from Horizontal compression. Please refer to vol1a **Memory Data Formats chapter -section** Media Memory Compression **for more details.**

Value	Name
0	Horizontal Compression Mode

Programming Notes
This bit is not used unless Memory Compression Enable is set to "1"

All Codec except JPEG Vertical Compression Only

Video Mode	Compression Mode
AVC Frame Only (No MBAFF or Field)	Horizontal
VP8 (Only Frame is supported	Horizontal
VC1 (No overlap smoothing, No field)	Not Supported
MPGE2 (No field)	Not Supported
JPEG	Not Supported

9 **Post Deblocking - Memory Compression Enable**

Format:	Enable

Memory compression will be attempted for this surface.

Value	Name
0	Compression Disable
1	Compression Enable

Programming Notes

Video Mode	Compression Enable
AVC Frame Only (No MBAFF or Field)	Yes
VP8 (Only Frame is supported	Yes
VC1 (No overlap smoothing, No field)	Yes

		MPGE2 (No field) | Yes JPEG | No (In JPEG mode, this surface is not used)
	8:7	**Post Deblocking - Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. Value | Name 00b | Highest priority 01b | Second highest priority 10b | Third highest priority 11b | Lowest priority
	6:0	**Post Deblocking - Memory Object Control State** Format: | MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).
7	31:6	**Original Uncompressed Picture Source Address** Format: | GraphicsAddress[31:6] Specifies the 64 byte aligned frame buffer address for fetching YUV pixel data from the original uncompressed input picture for encoding. This field is only valid in **encoding** mode.
	5:0	**Reserved** Format: | MBZ
8	31:16	**Reserved** Format: | MBZ
	15:0	**Original Uncompressed Picture Source Address High** Format: | GraphicsAddress[47:32] This field is for the upper range of Original Uncompressed Picture Source Address. This field is valid for **encoding** mode only.
9	31:15	**Reserved** Format: | MBZ
	14:13	**Original Uncompressed Picture - Tiled Resource Mode** **For Media Surfaces:** This field specifies the tiled resource mode.

MFX_PIPE_BUF_ADDR_STATE

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

12:11 **Reserved**

Format:	MBZ

10 **Original Uncompressed Picture - Memory Compression Mode**

Description
Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats chapter -section Media Memory Compression for more details. Note: This bit is not used unless Memory Compression Enable is set to "1" Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before

Value	Name
0	Horizontal Compression Mode
1	Vertical Compression Mode

9 **Original Uncompressed Picture - Memory Compression Enable**

Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before.

Value	Name
0	Compression Disable

8:7 **Original Uncompressed Picture Source - Arbitration Priority Control**

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

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

6:0 **Original Uncompressed Picture Source - Memory Object Control State**

Format:	MEMORY_OBJECT_CONTROL_STATE

Specifies the memory object control state for this surface and the associated Auxiliary surface

		(if any).
10	31:6	**StreamOut Data Destination Base Address** Format: GraphicsAddress[31:6] Specifies the 64 byte aligned address for outputting the per-MB indirect data to memory when **StreamOutEnable** is set in the MFX_PIPE_MODE_SELECT command. For Decoder : This field is used for transcoding purpose. For Encoder : This field is used for dynamic repeat of frame in PAK for Rate Control. Also used for feeding coding information back to the Host, Video Preprocessing Unit and ENC Unit. All data are written in fixed formats, and therefore all record sizes are known in the hardware. Hardware can calculate the offset into this base address for per-MB data.
	5:0	**Reserved** Format: MBZ
11	31:16	**Reserved** Format: MBZ
	15:0	**StreamOut Data Destination Base Address High** Format: GraphicsAddress[47:32] This field is for the upper range of Original Uncompressed Picture Source Address
12	31:15	**Reserved** Format: MBZ
	14:13	**StreamOut Data Destination - Tiled Resource Mode** **For Media Surfaces:** This field specifies the tiled resource mode. Value / Name / Description: 0h / TRMODE_NONE / No tiled resource 1h / TRMODE_TILEYF / 4KB tiled resources 2h / TRMODE_TILEYS / 64KB tiled resources 3h / Reserved /
	12:11	**Reserved** Format: MBZ
	10	**StreamOut Data Destination - Memory Compression Mode** Description

		Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats chapter -section Media Memory Compression for more details. Note: This bit is not used unless Memory Compression Enable is set to "1" Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before Only horizontal mode is supported. **Value** / **Name** 0 – Horizontal Compression Mode 1 – Vertical Compression Mode
	9	**StreamOut Data Destination - Memory Compression Enable** Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before. **Value** / **Name** 0 – Compression Disable
	8:7	**StreamOut Data Destination - Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. **Value** / **Name** 00b – Highest priority 01b – Second highest priority 10b – Third highest priority 11b – Lowest priority
	6:0	**StreamOut Data Destination - Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).
13	31:6	**Intra Row Store Scratch Buffer Base Address** Format: GraphicsAddress[31:6] This field provides the base address of the scratch buffer (read/write) used by the AVC/VP8 IntraPrediction unit to store MB information of the previous row for processing of each macroblock in the current row. The Intra Row Store buffer must be 64-byte cacheline aligned. Hardware uses the horizontal address of the current macroblock to address the Intra Row Store. This field is ignored in MPEG2 and VC1 mode. Max 256 cachelines for 4K pixels (1 cacheline for either MBAFF or non-MBAFF)Intra Row Store Scratch Buffer - Arbitration Priority Control **Programming Notes** This is one of the four RowStore Scratch Buffers which can be programmed to use the internal

MFX_PIPE_BUF_ADDR_STATE

		Media Cache (total size 640 CacheLine). When Intra Row Store Scratch Buffer Cache Select is programmed to "1", this data will be stored inside MFX Media Internal Storage. Driver then needs to program this Base Address between 0 to 639, indicating starting cachelines address to Media Cache. Driver needs to make sure the whole buffer fits into MFX Media Internal Storage. *(Notes: 1 cacheline per MB, and the buffer needs to have enough space for 1 MB row).*
	5:0	**Reserved** Format: MBZ
14	31:16	**Reserved** Format: MBZ
	15:0	**Intra Row Store Scratch Buffer Base Address High** Format: GraphicsAddress[47:32] This field is for the upper range of Intra RowStore/Scratch Buffer Base Address This field is ignored in MPEG2 and VC1 mode.
15	31:15	**Reserved** Format: MBZ
	14:13	**Intra Row Store Scratch Buffer - Tiled Resource Mode** **For Media Surfaces:** This field specifies the tiled resource mode.

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

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

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

	11	**Reserved** Format: MBZ
	10	**Intra Row Store Scratch Buffer - Memory Compression Mode**

Description
Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats chapter -section Media Memory Compression for more details. Note: This bit is not used unless Memory Compression Enable is set to "1"

Value	Name
0	Horizontal Compression Mode
1	Vertical Compression Mode

9 **Intra Row Store Scratch Buffer - Memory Compression Enable**

Value	Name
0	Compression Disable

Programming Notes
This surface is linear surface. This bit must be set to "0" since only TileY/TileYf/TileYs surface is allowed to be compressed

8:7 **Intra Row Store Scratch Buffer - Arbitration Priority Control**

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

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

6:0 **Intra Row Store Scratch Buffer - Memory Object Control State**

Format:	MEMORY_OBJECT_CONTROL_STATE

Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).

16 31:6 **Deblocking Filter Row Store Scratch Base Address**

Format:	GraphicsAddress[31:6]

Deblocking Filter Row Store is needed for:

- AVC and VC1 In-Loop Deblocking Filter
- VC1 Overlap-smoothing Filter

MFX_PIPE_BUF_ADDR_STATE

- AVC, VC1, and MPEG-2 Out-Of-Loop Deblocking Filter (Intel extension)

This field provides the 64 byte aligned base address of the scratch buffer (read and write) used by the deblocking filter unit to store MB information of the previous row for filtering of each macroblock in the current row. The Deblocking Filter Row Store buffer must be 64-byte cacheline aligned. Hardware uses the horizontal address of the current macroblock to address the Deblocking Filter Row Store. Max 6 cachelines for VC1 and MPEG2, and max 4 for AVC (for MBAFF, 2 for non-MBAFF)

Programming Notes

This is one of the four RowStore Scratch Buffers which can programmed to use the internal Media Cache (total size 640 CacheLine). When Deblocking Filter Row Store Scratch Buffer Cache Select is programmed to "1", this will be stored inside MFX Media Internal Storage. Driver then needs to program this Base Address between 0 to 639, indicating starting cachelines address location for this buffer. Driver needs to make sure the whole buffer fits into Media Internal Storage.
(Notes: 2 cachelines per MB for non-mbaff; 4 cahcelines per MB pair for mbaff, and the buffer needs to have enough space for 1 MB (pair) row).

5:0 Reserved

Format:	MBZ

17 — 31:16 Reserved

Format:	MBZ

15:0 Deblocking Filter Row Store Scratch Base Address High

Format:	GraphicsAddress[47:32]

This field is for the upper range of Deblocking Filter Row Store Scratch Buffer Address.

18 — 31:15 Reserved

Format:	MBZ

14:13 Deblocking Filter Row Store - Tiled Resource Mode

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

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

12 Deblocking Filter Row Store Scratch Buffer Cache Select

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

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

11 **Reserved**

Format:	MBZ

10 **Deblocking Filter Row Store Scratch - Memory Compression Mode**

Description
Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats chapter -section Media Memory Compression for more details. Note: This bit is not used unless Memory Compression Enable is set to "1" Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before.

Value	Name
0	Horizontal Compression Mode
1	Vertical Compression Mode

9 **Deblocking Filter Row Store Scratch - Memory Compression Enable**

Value	Name
0	Compression Disable

Programming Notes
This surface is linear surface. This bit must be set to "0" since only TileY/TileYf/TileYs surface is allowed to be compressed

8:7 **Deblocking Filter Row Store Scratch - Arbitration Priority Control**

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

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

6:0 **Deblocking Filter Row Store Scratch - Memory Object Control State**

		Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).
19..50	1023:0	**Reference Picture Base Addr** Format: REFERENCE_PICTURE_BASE_ADDR[16]
51	31:15	**Reserved** Format: MBZ
	14:13	**Reference Picture - Tiled Resource Mode** **For Media Surfaces:** This field specifies the tiled resource mode. 0h TRMODE_NONE — No tiled resource 1h TRMODE_TILEYF — 4KB tiled resources 2h TRMODE_TILEYS — 64KB tiled resources 3h Reserved
	12:9	**Reserved** Format: MBZ
	8:7	**Reference Picture - Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. 00b Highest priority 01b Second highest priority 10b Third highest priority 11b Lowest priority
	6:0	**Reference Picture - Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).
52	31:6	**Macroblock Buffer Base Address or Decoded Picture Error/Status Buffer Base Address** Format: GraphicsAddress[31:6] **For decoder:** Specifies the 64 byte aligned buffer address for writing a single error/status record into the memory when **Pic Error/Status Report Enable** is set in the

MFX_PIPE_BUF_ADDR_STATE

		MFX_PIPE_MODE_SELECT Command. The error/status record is written by HW at the end of decoding one single picture. The record is written in a fixed format, total 96-bits in size always. Please refer to "Media VDBOX -> Video Codec -> Other Codec Functions -> MFX Error Handling -> Decoder" session for the output format. **For encoder:** Specifies the 64 byte aligned buffer address for reading the per-MB indirect data from memory when **MacroblockStatEnable** is set in the MFX_AVC_IMG_STATE Command. This field is used for dynamic repeat of frame in PAK for Rate Control. Also used for feeding coding information back to the Host, Video Preprocessing Unit, and ENC Unit. All data are written in fixed formats, and therefore all record sizes are known in the hardware. Hardware can calculate the offset into this base address for per-MB data.
	5:0	**Reserved** Format: MBZ
53	31:16	**Reserved** Format: MBZ
	15:0	**Macroblock Buffer Base Address or Decoded Picture Error/Status Buffer Base Address High** Format: GraphicsAddress[47:32] This field is for the upper range of Macroblock Status Buffer Base Address
54	31:15	**Reserved** Format: MBZ
	14:13	**Macroblock Status Buffer - Tiled Resource Mode** **For Media Surfaces:** This field specifies the tiled resource mode.

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

	12:11	**Reserved** Format: MBZ
	10	**Macroblock Status Buffer - Memory Compression Mode**

Description
Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data

MFX_PIPE_BUF_ADDR_STATE

		Formats chapter -section Media Memory Compression for more details. Note: This bit is not used unless Memory Compression Enable is set to "1" Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before. **Value** / **Name** 0 – Horizontal Compression Mode 1 – Vertical Compression Mode
	9	**Macroblock Status Buffer - Memory Compression Enable** **Value** / **Name** 0 – Compression Disable **Programming Notes** This surface is linear surface. This bit must be set to "0" since only TileY/TileYf/TileYs surface is allowed to be compressed
	8:7	**Macroblock Status Buffer - Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. **Value** / **Name** 00b – Highest priority 01b – Second highest priority 10b – Third highest priority 11b – Lowest priority
	6:0	**Macroblock Status Buffer - Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).
55	31:6	**Macroblock ILDB StreamOut Buffer Base Address** Format: GraphicsAddress[31:6] Specifies the 64 byte aligned buffer address for writing MB ILDB parameter per MB to memory when **Debocker streamout enable** is set in the MFX_PIPE_MODE_SELECT Command. The ildb MB control parameters are written by HW at the end of each decoding MB. Only AVC edge information is being streamed out. It is used in AVC decode mode only.
	5:0	**Reserved**

		Format: MBZ
56	31:16	**Reserved** Format: MBZ
	15:0	**Macroblock ILDB StreamOut Buffer Base Address High** Format: GraphicsAddress[47:32] This field is for the upper range of Deblocking Filter Row Store Scratch Address
57	31:15	**Reserved** Format: MBZ
	14:13	**Macroblock ILDB StreamOut - Tiled Resource Mode** **For Media Surfaces:** This field specifies the tiled resource mode.

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

	12:11	**Reserved** Format: MBZ
	10	**Macroblock ILDB StreamOut Buffer - Memory Compression Mode**

Description
Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats chapter -section Media Memory Compression for more details. Note: This bit is not used unless Memory Compression Enable is set to "1" Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before.

Value	Name
0	Horizontal Compression Mode
1	Vertical Compression Mode

	9	**Macroblock ILDB StreamOut Buffer - Memory Compression Enable**

Value	Name
0	Compression Disable

MFX_PIPE_BUF_ADDR_STATE

		Programming Notes This surface is linear surface. This bit must be set to "0" since only TileY/TileYf/TileYs surface is allowed to be compressed
	8:7	**Macroblock ILDB StreamOut Buffer - Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. **Value** / **Name** 00b – Highest priority 01b – Second highest priority 10b – Third highest priority 11b – Lowest priority
	6:0	**Macroblock ILDB StreamOut Buffer - Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).
58	31:6	**Second Macroblock ILDB StreamOut Buffer Base Address** Format: GraphicsAddress[31:6] 64 byte aligned buffer. Specifies the 64 byte aligned buffer address for writing MB ILDB parameter per MB to memory when Debocker streamout enable is set in the MFX_PIPE_MODE_SELECT Command. The ildb MB control parameters are written by HW at the end of each decoding MB. Only AVC edge information is being streamed out. It is used in AVC decode mode only.
	5:0	**Reserved** Format: MBZ
59	31:16	**Reserved** Format: MBZ
	15:0	**Second Macroblock ILDB StreamOut Buffer Base Address High** Format: GraphicsAddress[47:32] This field is for the upper range of Second Macroblock ILDB StreamOutBuffer Base Address.
60	31:15	**Reserved** Format: MBZ

MFX_PIPE_BUF_ADDR_STATE

	14:13	**Second Macroblock ILDB StreamOut Buffer - Tiled Resource Mode** **For Media Surfaces:** This field specifies the tiled resource mode. 0h: TRMODE_NONE — No tiled resource 1h: TRMODE_TILEYF — 4KB tiled resources 2h: TRMODE_TILEYS — 64KB tiled resources 3h: Reserved
	12:11	**Reserved** Format: MBZ
	10	**Second Macroblock ILDB StreamOut Buffer - Memory Compression Mode** **Description** Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats chapter -section Media Memory Compression for more details. Note: This bit is not used unless Memory Compression Enable is set to "1" Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before. 0: Horizontal Compression Mode 1: Vertical Compression Mode
	9	**Second Macroblock ILDB StreamOut Buffer - Memory Compression Enable** 0: Compression Disable **Programming Notes** This surface is linear surface. This bit must be set to "0" since only TileY/TileYf/TileYs surface is allowed to be compressed
	8:7	**Second Macroblock ILDB StreamOut Buffer - Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.
	6:0	**Second Macroblock ILDB StreamOut Buffer - Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE

MFX_PIPE_BUF_ADDR_STATE

		Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).
61	31	**Reference Picture 15 - Memory Compression Mode** Description: Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats chapter -section Media Memory Compression for more details. Note: This bit is not used unless Memory Compression Enable is set to "1" Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before. 0: Horizontal Compression Mode 1: Vertical Compression Mode
	30	**Reference Picture 15 - Memory Compression Enable** 0: Compression Disable 1: Compression Enable
	29	**Reference Picture 14 - Memory Compression Mode** Description: Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats section Media Memory Compression for more details. Note: This bit is not used unless Memory Compression Enable is set to "1" Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before. 0: Horizontal Compression Mode 1: Vertical Compression Mode
	28	**Reference Picture 14 - Memory Compression Enable** 0: Compression Disable 1: Compression Enable

27 **Reference Picture 13 - Memory Compression Mode**

Description
Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats section Media Memory Compression for more details. Note: This bit is not used unless Memory Compression Enable is set to "1" Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before.

Value	Name
0	Horizontal Compression Mode
1	Vertical Compression Mode

26 **Reference Picture 13 - Memory Compression Enable**

Value	Name
0	Compression Disable
1	Compression Enable

25 **Reference Picture 12 - Memory Compression Mode**

Description
Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats section Media Memory Compression for more details. Note: This bit is not used unless Memory Compression Enable is set to "1" Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before.

Value	Name
0	Horizontal Compression Mode
1	Vertical Compression Mode

24 **Reference Picture 12 - Memory Compression Enable**

Value	Name
0	Compression Disable
1	Compression Enable

23 **Reference Picture 11 - Memory Compression Mode**

Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data

Formats chapter -section Media Memory Compression for more details. Note: This bit is not used unless Memory Compression Enable is set to "1" Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before.

Value	Name
0	Horizontal Compression Mode
1	Vertical Compression Mode

22 **Reference Picture 11 - Memory Compression Enable**

Value	Name
0	Compression Disable
1	Compression Enable

21 **Reference Picture 10 - Memory Compression Mode**

Description
Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats section Media Memory Compression for more details. Note: This bit is not used unless Memory Compression Enable is set to "1" Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before.

Value	Name
0	Horizontal Compression Mode
1	Vertical Compression Mode

20 **Reference Picture 10 - Memory Compression Enable**

Value	Name
0	Compression Disable
1	Compression Enable

19 **Reference Picture 9 - Memory Compression Mode**

Description
Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats section Media Memory Compression for more details. Note: This bit is not used unless Memory Compression Enable is set to "1" Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before.

Value	Name
0	Horizontal Compression Mode
1	Vertical Compression Mode

18 **Reference Picture 9 - Memory Compression Enable**

Value	Name
0	Compression Disable
1	Compression Enable

17 **Reference Picture 8 - Memory Compression Mode**

Description
Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats section Media Memory Compression for more details. Note: This bit is not used unless Memory Compression Enable is set to "1" Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before.

Value	Name
0	Horizontal Compression Mode
1	Vertical Compression Mode

16 **Reference Picture 8 - Memory Compression Enable**

Value	Name
0	Compression Disable
1	Compression Enable

15 **Reference Picture 7 - Memory Compression Mode**

Description
Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats section Media Memory Compression for more details. Note: This bit is not used unless Memory Compression Enable is set to "1" Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before.

Value	Name
0	Horizontal Compression Mode

1	Vertical Compression Mode

14 Reference Picture 7 - Memory Compression Enable

Value	Name
0	Compression Disable
1	Compression Enable

13 Reference Picture 6 - Memory Compression Mode

Description
Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats section Media Memory Compression for more details. Note: This bit is not used unless Memory Compression Enable is set to "1" Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before.

Value	Name
0	Horizontal Compression Mode
1	Vertical Compression Mode

12 Reference Picture 6 - Memory Compression Enable

Value	Name
0	Compression Disable
1	Compression Enable

11 Reference Picture 5 - Memory Compression Mode

Description
Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats chapter -section Media Memory Compression for more details. Note: This bit is not used unless Memory Compression Enable is set to "1" Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before.

Value	Name
0	Horizontal Compression Mode
1	Vertical Compression Mode

10 Reference Picture 5 - Memory Compression Enable

Value	Name
0	Compression Disable
1	Compression Enable

9 **Reference Picture 4 - Memory Compression Mode**

Description
Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats section Media Memory Compression for more details. Note: This bit is not used unless Memory Compression Enable is set to "1" Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before.

Value	Name
0	Horizontal Compression Mode
1	Vertical Compression Mode

8 **Reference Picture 4 - Memory Compression Enable**

Value	Name
0	Compression Disable
1	Compression Enable

7 **Reference Picture 3 - Memory Compression Mode**

Description
Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats chapter -section Media Memory Compression for more details. Note: This bit is not used unless Memory Compression Enable is set to "1" Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before.

Value	Name
0	Horizontal Compression Mode
1	Vertical Compression Mode

6 **Reference Picture 3 - Memory Compression Enable**

MFX_PIPE_BUF_ADDR_STATE

Value	Name
0	Compression Disable
1	Compression Enable

5 **Reference Picture 2 - Memory Compression Mode**

Description
Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats section Media Memory Compression for more details. Note: This bit is not used unless Memory Compression Enable is set to "1" Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before.

Value	Name
0	Horizontal Compression Mode
1	Vertical Compression Mode

4 **Reference Picture 2 - Memory Compression Enable**

Value	Name
0	Compression Disable
1	Compression Enable

3 **Reference Picture 1 - Memory Compression Mode**

Description
Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats chapter -section Media Memory Compression for more details. Note: This bit is not used unless Memory Compression Enable is set to "1" Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before.

Value	Name
0	Horizontal Compression Mode
1	Vertical Compression Mode

2 **Reference Picture 1 - Memory Compression Enable**

Value	Name
0	Compression Disable

		1 Compression Enable
	1	**Reference Picture 0 - Memory Compression Mode** Description: Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats -section Media Memory Compression for more details. Note: This bit is not used unless Memory Compression Enable is set to "1" Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before. Value 0: Horizontal Compression Mode Value 1: Vertical Compression Mode
	0	**Reference Picture 0 - Memory Compression Enable** Value 0: Compression Disable Value 1: Compression Enable
62	31:6	**Scaled Reference Surface Base Address** Format: GraphicsAddress[31:6] Specifies the 64 byte aligned down scaled reference frame buffer addresses that needs to be used by the PAK down-scaler to write the down scaled pixels. Only the luma pixels will be downscaled and written to the surface
	5:0	**Reserved** Format: MBZ
63	31:16	**Reserved** Format: MBZ
	15:0	**Scaled Reference Surface Base Address High** Format: GraphicsAddress[47:32] This field is for the upper range of Scaled Reference Surface Base Address.
64	31:15	**Reserved**

Format:	MBZ

14:13 Scaled Reference Surface - Tiled Resource Mode

For Media Surfaces: This field specifies the tiled resource mode

Value	Name	Description
0h	TRMODE_NONE	No tiled resource
1h	TRMODE_TILEYF	No tiled resource
2h	TRMODE_TILEYS	No tiled resource
3h	Reserved	

12:11 Reserved

Format:	MBZ

10 Scaled Reference Surface - Memory Compression Mode

Description
Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats chapter -section Media Memory Compression for more details.

Value	Name
0	Horizontal Compression Mode
1	Vertical Compression Mode

9 Scaled Reference Surface - Memory Compression Enable

Format:	Enable

Memory compression shouldn't be enabled for this surface.

8:7 Scale Reference Surface - Arbitration Priority Control

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

Value	Name
00b	Highest Priority
01b	Second Highest Priority
10b	Third Highest Priority
11b	Lowest Priority

6:0 Scaled Reference Surface - Memory Object Control State

		Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).
65	31:6	**SliceSize StreamOut Data Destination Base Address** Format: GraphicsAddress[31:6] Specifies the 64 byte aligned Slice Size streamout surface address. Here slice sizes are written out. This surface can be used to determine the slice start location.
	5:0	**Reserved** Format: MBZ
66	31:16	**Reserved** Format: MBZ
	15:0	**SliceSize StreamOut Data Destination Base Address High** Format: GraphicsAddress[47:32] This field is for the upper range of Slice Size Streamout Surface Base Address.
67	31:15	**Reserved** Format: MBZ
	14:13	**SliceSize StreamOut Data Destination - Tiled Resource Mode** For Media Surfaces: This Surface is never tiled. Value / Name / Description: 0h / TRMODE_NONE / No tiled resource 1h / TRMODE_TILEYF / No tiled resource 2h / TRMODE_TILEYS / No tiled resource 3h / Reserved /
	12:11	**Reserved** Format: MBZ
	10	**SliceSize StreamOut Data Destination - Memory Compression Mode**

Description
Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats chapter -section Media Memory Compression for more details.

Value	Name
0	Horizontal Compression Mode
1	Vertical Compression Mode

9	**SliceSize StreamOut Data Destination - Memory Compression Enable**

Format:	Enable

Memory compression is never enabled for this surface

8:7	**SliceSize StreamOut Data Destination - Arbitration Priority Control**

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

Value	Name
00b	Highest Priority
01b	Second Highest Priority
10b	Third Highest Priority
11b	Lowest Priority

6:0	**SliceSize StreamOut Data Destination - Memory Object Control State**

Format:	MEMORY_OBJECT_CONTROL_STATE

Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).

MFX_PIPE_MODE_SELECT

MFX_PIPE_MODE_SELECT

Source: VideoCS

Length Bias: 2

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

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

	Format:	AES_CONTROL

23:19	**Reserved**	
	Format:	MBZ

18	**Extended stream out enable**	
	Format:	U1

This bit can be set only when VDEnc_Mode is set.
When this bit is set and MB stream out is enabled, per MB 1CL of data is streamed out. The actual contents of the stream out are listed in Media VDBOX > Encoder VDEnc mode StreamOut Data Structure Definition.
When this bit is not set, per MB ¼ CL data is streamed out. The actual contents of the stream out are listed in Media VDBOX > Encoder StreamOut Mode Data Structure Definition.

17 **Decoder Short Format Mode**

For IT mode, this bit must be 0.

Value	Name	Description
0	Short Format Driver Interface **[Default]**	AVC/VC1/MVC/VP8 Short Format Mode is in use **Note: There is no Short Format for VP8 yet, so this field must be set to 1 for VP8.**
1	Long Format Driver Interface	[] AVC/VC1/MVC/VP8 Long 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
2h	Deblocker Mode	Configure the MFD Engine for Standalone Deblocker Mode. Require streamout AVC edge control information from preceding decoding pass.
3h	Reserved	

14 **Standalone VDEnc_Mode Enable**

Format:	Enable

This field indicates to PAK if this is standalone VDEnc mode. This is primarily a validation mode.

Value	Name
0h	VDEnc+PAK
1h	PAK Only

13 **VDEnc_Mode**

This field indicates if PAK is working in legacy MBEnc mode or the VDEnc mode.

Value	Name	Description
0h	MBEnc mode	PAK is working in legacy mode
1h	VDEnc mode	PAK is working in VDEnc mode

12 **Deblocker Stream-Out Enable**

This field indicates if Deblocker information is going to be streamout during VLD decoding. For AVC, it is needed to enable the deblocker streamout as the AVC Disable_DLKFilterIdc is a slice level parameters. Driver needs to determine ahead of time if at least one slice of the current frame/ has deblocker ON.

Value	Name
0h	Disable
1h	Enable

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 pictures; otherwise, hardware might overwrite previous written data if driver does not read it fast enough In encoder modes: Not used
Please refer to "Media VDBOX -> Video Codec -> Other Codec Functions -> MFX Error Handling -> Decoder" session for the output format.

Value	Name
0h	Disable
1h	Enable

10 **Stream-Out Enable**

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

Value	Name
0h	Disable
1h	Enable

Programming Notes
In decoder modes: The Stream-Out feature is added to support transcoding. While decoding the input compressed stream, selected decoded information may be used by the encoder for re-compression. In encoder modes: This feature used to perform dynamic Multipass of PAK for

conformance purpose. Also it provides feedback to host (ENC) for future needs. Software can use this bit to disable writing PAK steam data to the streamout buffer for last pass of frame in PAK. Thus, save memory bandwidth.

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

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

Value	Name
0h	Disable
1h	Enable

8 **Pre Deblocking Output Enable (PreDeblockOutEnable)**

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

Value	Name
0h	Disable
1h	Enable

7 **Scaled Surface Enable**

This field indicates if the scaled surface is enabled. This field enables the 4x HME downscalar of the reconstructed image. Only supported for AVC and VP8 formats.

Value	Name
0h	Disable
1h	Enable

6 **Frame Statistics StreamOut Enable**

This field controls the frame level statistics streamout from the PAK.

Note: This field needs to be always "Enabled" in VD_Enc mode. In case of non-VDEnc mode, this can be used to control the frame statistics output from the PAK.

Value	Name
0h	Disable
1h	Enable

5 **Stitch Mode**

Exists If:	//CodecSel=Encode and StandardSel=AVC

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

4 **Codec Select**

Value	Name	Description

		0h	Decode	
		1h	Encode	Valid only if StandardSel is AVC and MPEG2)

	3:0	**Standard Select**		
		Value	**Name**	**Description**
		0000b	MPEG2	
		0001b	VC1	
		0010b	AVC	Covers both AVC and MVC
		0011b	JPEG	
		0101b	VP8	Decoder, Encoder
		0110b	Reserved	
		0111b	Reserved	
		1111b	UVLD	SW decoder w/ embedded micro-controller and co-processor
2	31:0	**Reserved**		
		Format:		MBZ
3	31:0	**Pic Status/Error Report ID**		
		Exists If:	//Decoder Mode Only	
		Format:	U32	
		In decoder modes: Error reporting is written out once per frame. This field along with the VLD error status bits are packed into one cache and written to the memory location specified by "Decoded Picture Error/Status Buffer address" listed in the MFX_PIPE_BUF_ADDR_STATE Command.		
		Value	**Name**	**Description**
		0h	32-bit unsigned	Unique ID Number
		1h	Reserved	
4	31:0	**Reserved**		
		Format:		MBZ

MFX_QM_STATE

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**

MFX_QM_STATE

		Format: MBZ
	1:0	**AVC** Exists If: //AVC- Decoder Only **For AVC QM Type**: This field specifies which Quantizer Matrix is loaded. 0 – AVC_4x4_Intra_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs) 1 – AVC_4x4_Inter_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs) 2 – AVC_8x8_Intra_MATRIX 3 – AVC_8x8_Inter_MATRIX
	1:0	**MPEG2** Exists If: //MPEG2- Decoder Only **For MPEG2 QM Type**: This field specifies which Quantizer Matrix is loaded. 0 – MPEG_INTRA_QUANTIZER_MATRIX 1 – MPEG_NON_INTRA_QUANTIZER_MATRIX 2-3 – Reserved
	1:0	**JPEG** Exists If: //JPEG- Encoder Only **For JPEG QM Type**:This field specifies which Quantizer Matrix is loaded. 0 – JPEG_Luma_Y_QUANTIZER_MATRIX (or R) 1 – JPEG_Chroma_Cb_QUANTIZER_MATRIX (or G) 2 – JPEG_Chroma_Cr_QUANTIZER_MATRIX (or B) **Programming Notes** For JPEG encoder, each quantization element presents 16-bit 1/QM[i][j]. In RGB encoding, because the order input image components can be RGB, GBR, BGR, YUV, the value 0 is used for the first image component, the value 1 is used for the second image component, and the value 2 is used for the third image component.
2..33	1023:0	**Forward Quantizer Matrix** The format of a Quantizer Matrix is an 8x8 matrix in raster order. Each element is an unsigned byte.

MFX_STATE_POINTER

MFX_STATE_POINTER

Source: VideoCS

Length Bias: 2

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

The indirection is designed to facilitate context switching in the middle of a codec operation. The smallest granularity of interruption is designed to be at a completed MB row in AVC/VC1/MPEG2 IT and AVC PAK operating modes as well as in VC1/MPEG2 VLD mode. There is no support for context switch in AVC VLD mode. Hardware supports up to 4 separate indirect state pointers, allowing software to manage the grouping of state commands. During context switch, hardware restores (re-issues) the latest version of each indirect state pointer, if present.

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFX_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:24	**Media Command Opcode** Default Value: 0h MFX_COMMON_STATE Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 6h Format: OpCode
	15:12	**Reserved**

MFX_STATE_POINTER

		Format: MBZ
	11:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n Total Length - 2
1	31:5	**State Pointer** Format: GeneralStateOffset[31:5]Indirect State Buffer Specifies the 32-byte aligned address of an Indirect State Buffer. This pointer is relative to the General State Base Address.
	4:2	**Reserved** Format: MBZ
	1:0	**State Pointer Index** Specifies one of the four indirect state pointers to program.

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

MFX_STITCH_OBJECT

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. Context switch interrupt is not supported by this command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFC_STITCH_OBJECT Format: OpCode
	26:24	**Media Command Opcode** Default Value: 0h MFX_COMMON Format: OpCode
	23:21	**SubOpcode A** Default Value: 2h Format: OpCode
	20:16	**SubOpcode B** Default Value: Ah Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) = Variable Length in DW (>= 3) Format: =n Total Length - 2 If it is 3, it indicates the absent of inline data.
1	31:18	**Reserved**

		Format: MBZ
	17:16	**Source Data Starting Byte Offset** Source Data Starting Byte Position within the very first inline DW.
	15:14	**Reserved** Format: MBZ
	13:8	**Source Data Ending Bit Inclusion** Source Data to be included in the very last inline DW. Follows the MSBit is the upper bit of each byte within the DW. The lower byte is actually processed first.For example, SrCDataEndingBitInclusion =9, bit 7:0 and bit 15 are included as valid header data. Value: [1,32] — Name:
	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

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 picture (becoming reference pictures as well for subsequent temporal inter-prediction)

Since there is only one media surface state being active during the entire encoding/decoding process, all the uncompressed/reconstructed pictures are defined to have the same surface state. The primary difference among picture surface states is their individual programmed base addresses, which are provided by other state commands and not included in this command. MFX engine is making the association of surface states and corresponding buffer base addresses.
MFX engine currently supports only one media surface type for video and that is the NV12 (Planar YUV420 with interleaved U (Cb) and V (Cr). For optimizing memory efficiency based on access patterns, only TileY is supported. For JPEG decoder, only IMC1 and IMC3 are supported. Pitch can be wider than the Picture Width in pixels and garbage will be there at the end of each line. The following describes all the different formats that are supported and not supported in Gen7 MFX :

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

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

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

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

MFX_SURFACE_STATE

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

Programming Notes

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

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

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

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

MFX_SURFACE_STATE

		Format: MBZ
	11:0	**DWord Length** Format: =n Total Length - 2 **Value** / **Name** / **Description** 4h / DWORD_COUNT_n **[Default]** / Excludes DWord (0,1)
1	31:4	**Reserved** Format: MBZ
	3:0	**Surface Id** Format: U4 **Value** / **Name** / **Description** 0100b / Source Input Picture (encoder) / 8-bit uncompressed data 0101b / Reconstructed Scaled Reference Picture / 8-bit data
2	31:18	**Height** Format: U14-1 Height **Description** This field specifies the height of the Picture in units of pixels/residuals. For PLANAR surface formats, this field indicates the height of the Y (luma) plane. Note : Gen7 Video Codecs must program less than and equal to 4K.(In future, it will be ideal to have this field define in a WORD boundary.)AVC - multiple of 2 MB rows for field pictureVC1 - multiple of 4 pixels for field pictureMPEG2 - multiple of 2 MB rows for field picJPEG - multiple 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.
	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.
	3:2	**Reserved** Format: MBZ
	1:0	**Cr(V)/Cb(U) Pixel Offset V Direction** Format: U0.2 exactly as shown in the original spec Specifies the distance to the U/V values with respect to the even numbered Y channels in the V direction **Programming Notes** This field is ignored for all formats except PLANAR_420_8
3	31:28	**Surface Format** Format: U4 Specifies the format of the surface. All of the Y and G channels will use table 0 and all of the Cr/Cb/R/B channels will use table 1.Usage: For 420 planar YUV surface, use 4; for monochrome surfaces, use 12. For monochrome surfaces, hardware ignores control fields for Chroma planes. This field must be set to 4 - PLANAR_420_8, or 12 - Y8_UNORMNot used for MFX, and is ignored. But for JPEG decoding, this field should be programmed to the same format as JPEG_PIC_STATE. For video codec, it should set to 4 always.

Value	Name	Description
0	YCRCB_NORMAL	
1	YCRCB_SWAPUVY	
2	YCRCB_SWAPUV	
3	YCRCB_SWAPY	
4	PLANAR_420_8	(NV12, IMC1,2,3,4, YV12)
5	PLANAR_411_8	Deinterlace Only
6	PLANAR_422_8	Deinterlace Only

7	STMM_DN_STATISTICS	Deinterlace Only
8	R10G10B10A2_UNORM	Sample_8x8 Only
9	R8G8B8A8_UNORM	Sample_8x8 Only
10	R8B8_UNORM (CrCb	Sample_8x8 Only
11	R8_UNORM (Cr/Cb)	Sample_8x8 Only
12	Y8_UNORM	Sample_8x8 Only

27 **Interleave Chroma**

Format:	Enable

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

Value	Name
1	Enable
0	Disable

26:20 **Reserved**

Format:	MBZ

19:3 **Surface Pitch**

Format:	U17-1 pitch in Bytes

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

Value	Name
[0,131071]	

Programming Notes

For tiled surfaces, the pitch must be a multiple of the tile width (i.e.128 bytes aligned). If Half Pitch for Chroma is set, this field must be a multiple of two tile widths for tiled surfaces, or a multiple of 2 bytes for linear surfaces. For Y-tiled surfaces: Range = [127, 131071] to [128B,128KB] = [1 tile, 1024 tiles]

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

If Media Memory Compression is enabled, the following max pitch size restriction must be honored. For larger resolution, Media Memory compression Must be disabled.

Tiling	Pixel	Max Frame Width	Max Frame Width	Max Pitch

Mode	Format	(bytes)	(pixels)	(bytes)
Legacy 4K	8bpp	16k	16k	16k + 127
	16bpp	16k	8k	16k + 127
	32bpp	16k	4k	16k + 127
	64bpp	16k	2k	16k + 127
	128bpp	16k	1k	16k + 127
TileYF	8bpp	8k	8k	8k + 63
	16bpp	16k	8k	16k + 127
	32bpp	16k	4k	16k + 127
	64bpp	16k	2k	16k + 255
	128bpp	16k	1k	16k + 255
TileYS	8bpp	16k	16k	16k + 255
	16bpp	16k	8k	16k + 511
	32bpp	16k	4k	16k + 511
	64bpp	16k	2k	16k + 1023
	128bpp	16k	1k	16k + 1023

2 **Half Pitch for Chroma**

Format:	Enable

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

1 **Tiled Surface**

Format:	Boolean

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

Value	Name	Description
0	False	Linear
1	True	Tiled

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

0 **Tile Walk**

		(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.

Format:	3D_Tilewalk

Value	Name	Description
0h	XMAJOR	TILEWALK_XMAJOR
1h	YMAJOR	TILEWALK_YMAJOR

Programming Notes
The corresponding cache(s) must be invalidated before a previously accessed surface is accessed again with an altered state of this bit

4	31	**Reserved**

Format:	MBZ

	30:16	**X Offset for U(Cb)**

Format:	U15 Pixel Offset

This field specifies the horizontal offset in pixels from the Surface Base Address to the start (origin) of the U(Cb) plane or the interleaved UV plane if Interleave Chroma is enabled. This field is only used for PLANAR surface formats. This field must be set to zero.X Offset for U(Cb) in pixel (This field must be zero for NV12 and IMC 1 and 3)

Programming Notes
For PLANAR_420 and PLANAR_422 surface formats, this field must be zero.

	15	**Reserved**

Format:	MBZ

	14:0	**Y Offset for U(Cb)**

Format:	U15 Pixel Row Offset

This field specifies the vertical offset in rows from the Surface Base Address to the start (origin) of the U(Cb) plane or the interleaved UV plane if Interleave Chroma is enabled. This field is only used for PLANAR surface formats.

Programming Notes
For PLANAR_420 and PLANAR_422 surface formats, this field must be multiple of 16 pixels - i.e. multiple MBs. For JPEG, this field must be a multiple of 16 pixels.

5	31:29	**Reserved**

Format:	MBZ

	28:16	**X Offset for V(Cr)** Format: U13 Offset in Pixels This field must be zero for NV12 and IMC 1 and 3 This field specifies the horizontal offset in pixels from the Surface Base Address to the start (origin) of the V(Cr) plane. This field is only used for PLANAR surface formats with Interleave Chroma disabled. **Programming Notes** For PLANAR_420 and PLANAR_422 surface formats, this field must indicate an even number of pixels.
	15:0	**Y Offset for V(Cr)** Format: U16 Row Offset in Pixels This field specifies the vertical offset in rows from the Surface Base Address to the start (origin) of the V(Cr) plane. This field is only used for PLANAR surface formats with Interleave Chroma disabled. This field is ignored by all video codec, only used by JPEG. **Programming Notes** For PLANAR_420 surface formats, this field must be multiple of 16 pixels - i.e. multiple MBs. For JPEG, this field must be a multiple of 16 pixels.

MFX_VC1_DIRECTMODE_STATE

MFX_VC1_DIRECTMODE_STATE

Source: VideoCS

Length Bias: 2

Exists If: //VC1 decoding in VLD modes

This is a picture level command and should be issued only once, even for a multi-slices picture. There is only one DMV buffer for read (when processing a B-picture) and one for write (when processing a P-Picture). Each DMV record is 64 bits per MB, to store the top and bottom field MVs (32-bit MVx,y each).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_VC1_DIRECTMODE_STATE Format: OpCode
	26:24	**Media Command Opcode** Default Value: 2h VC1_COMMON Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 2h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 0005h Excludes DWord (0,1) Format: =n Total Length - 2
1..2	63:0	**Direct MV Write Buffer - Base Address** Format: **SplitBaseAddress64ByteAligned**

		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
3	31:0	**Direct MV Write Buffer - Attributes** Format: **MemoryAddressAttributes**
4..5	63:0	**Direct MV Reference Buffer - Base Address** Format: **SplitBaseAddress64ByteAligned** 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.
6	31:0	**Direct MV Reference Buffer - Attributes** Format: **MemoryAddressAttributes**

MFX_VC1_PRED_PIPE_STATE

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 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 performing 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

MFX_VC1_PRED_PIPE_STATE

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 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

MFX_VC1_PRED_PIPE_STATE

2	31:30	**Reserved** Format: MBZ
	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 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 - double -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 - double - 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 - double - 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.
4	31:30	**Reserved** Format: MBZ
	29:24	**LumShift2- single - BWD** 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.

MFX_VC1_PRED_PIPE_STATE

	23:22	**Reserved** Format: MBZ
	21:16	**LumShift1 - single - BWD** 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 - BWD** 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 - BWD** 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.
5	31:30	**Reserved** Format: MBZ
	29:24	**LumShift2- double - BWD** 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 - double -BWD** Format: U6

MFX_VC1_PRED_PIPE_STATE

		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 - double - BWD** 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 - double - BWD** 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.

MFX_VP8_BSP_BUF_BASE_ADDR_STATE

MFX_VP8_BSP_BUF_BASE_ADDR_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 Video Codec Format: OpCode
	26:24	**Media Command OpCode** Default Value: 4h VP8 Format: OpCode
	23:21	**Sub Opcode A** Default Value: 2h VP8 Common Format: OpCode
	20:16	**Sub Opcode B** Default Value: 3h MFX_VP8_BSP_BUF_BASE_ADDR_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Value / Name / Description 000h / Excludes DWord (0,1) **[Default]** / A special case to provide a dummy image state for stitch mode operation. In this case, fields in DW1 which is part of the dummy image state command are ignored by hardware." 008h / / Used for normal encode mode
1..2	63:0	**Frame Header - Base Address** Format: SplitBaseAddress64ByteAligned 64 byte aligned, 48-bit Abs. Address StreamIn Surface **Note:**The format is linear vs. tile for better performance.

MFX_VP8_BSP_BUF_BASE_ADDR_STATE

3	31:0	**Frame Header - Attributes** Format: **MemoryAddressAttributes**
4..5	63:0	**Intermediate Buffer - Base Address** Format: **SplitBaseAddress64ByteAligned** 64 byte aligned, 48-bit AbsAddr StreamIn Surface **Note:**The format is linear vs. tile for better performance.
6	31:0	**Intermediate Buffer - Attributes** Format: **MemoryAddressAttributes**
7..14	255:0	**Intermediate Buffer Partition Offset** **Programming Notes** All **Intermediate Buffer Partition-[i] Offset** (i = 1 to 8) and **Intermediate Buffer Max Size** need to be cacheline aligned (64Byte aligned).
15	31:0	**Intermediate Buffer Max Size** Format: U32
16..17	63:0	**Final Frame - Base Address** Format: **SplitBaseAddress64ByteAligned** 64 byte aligned, 48-bit AbsAddr StreamIn Surface **Note:**The format is linear vs. tile for better performance.
18	31:0	**Final Frame - Attributes** Format: **MemoryAddressAttributes**
19	31:6	**Reserved** Format: MBZ
	5:0	**Final Frame Byte Offset** Format: U6 Specify byte offset within a 64-byte cacheline where the bitstream should be inserted at.
20..21	63:0	**Streamout - Base Address** Format: **SplitBaseAddress64ByteAligned** 64 byte aligned,48-bit AbsAddr StreamIn Surface **Note:**The format is linear vs. tile for better performance.
22	31:0	**Streamout - Attributes** Format: **MemoryAddressAttributes**
23..24	63:0	**Coeff Probs StreamIn Surface - Base Address** Format: **SplitBaseAddress64ByteAligned**

MFX_VP8_BSP_BUF_BASE_ADDR_STATE

		64 byte aligned, 48-bit AbsAddr StreamIn Surface **Note:**The format is linear vs. tile for better performance.
25	31:0	**Coeff Probs StreamIn Surface - Attributes** Format: **MemoryAddressAttributes**
26..27	63:0	**Token Statistics Surface - Base Address** Format: **SplitBaseAddress64ByteAligned** 64 byte aligned, 48-bit Abs. Address StreamIn Surface **Note:**The format is linear vs. tile for better performance.
28	31:0	**Token Statistics Surface - Attributes** Format: **MemoryAddressAttributes**
29..30	63:0	**MPC RowStore Surface - Base Address** Format: **SplitBaseAddress64ByteAligned** Abs. Address StreamIn/StreamOut Surface. **Note:** The format is linear vs. tile for better performance. GraphicsAddress is a 64-bit value [63:0], but only a portion of it is used by hardware. The upper reserved bits are ignored and MBZ.
31	31:0	**MPC RowStore Surface - Attributes** Format: **MemoryAddressAttributes**

MFX_VP8_Encoder_CFG

MFX_VP8_Encoder_CFG

Source: VideoCS

Length Bias: 2

This must be the very first command to issue after the surface state, the pipe select and base address setting commands and must be issued before MFX_VP8_PIC_STATE.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Video Codec Format: OpCode
	26:24	**Media Command OpCode** Default Value: 4h VP8 Format: OpCode
	23:21	**Sub Opcode A** Default Value: 2h VP8 Common Format: OpCode
	20:16	**Sub Opcode B** Default Value: 1h MFX_VP8_ENCODER_CFG Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Value / Name / Description: 000h — Excludes DWord (0,1) **[Default]** — A special case to provide a dummy image state for stitch mode operation. In this case, fields in DW1 which is part of the dummy image state command are ignored by hardware." 01Dh — Used for normal encode mode
1	31:11	**Reserved** Format: MBZ

MFX_VP8_Encoder_CFG

	10	**VBSPunitPowerClock Gating Disable** Format: U1 VBSPunit Power Clock Gating Disable.
	9	**Compressed Bitstream Output Disable** Format: U1 Disable Compressed Bitstream Output.**(Both Final Bitstream and Intermediate bit buffer)**
	8	**Finer BRC Enable** Format: U1 Enable Finer BRC Feature.
	7	**Per Segment Delta Qindex / LoopFilter Disable** Format: U1 Disable Per Segment Delta Qindex / Loop Filter in Rate Control.
	6	**Rate Control Initial Pass** Format: U1 Value / Name: 1 – Initial pass 0 – Subsequence Pass(es)
	5	**Skip Final Bitstream when Over / Under flow** Format: U1 Skip Final Bitstream conditionally on Over/Under flow in rate control and intermediate Bit Buffer Overrun.
	4	**Update Segment Feature Data Flag** Exists If: //VP8 Encoder Format: U1 Enable for Frame Header per Segment Quantizer / LoopFilter Update
	3	**Bitstream Statistics Output Enable**

		Enable Bitstream Statistics Output at Memory Surface in MFX_VBSP_BUF_ADDR_STATE DW[26:28]
	2	**Token Statistics Output Enable** Enable Token Statistics Output at Memory Surface in MFX_VBSP_BUF_ADDR_STATE DW[26:28]
	1	**Final Bitstream Output Disable** Format: U1 Disable Final Bitstream Output.
	0	**Performance Counter Enable** Format: U1 Enable Performance Counter in Streamout.
2	31:8	**Reserved** Format: MBZ
	7	**Qindex_Clamp_High_mask for overflow** Format: U1 If current frame is overflow and this mask is set, it would mask out MFX_VP8_Img_Status register. DW1.bit1. In another word, subsequent passes would be skipped.
	6	**Qindex_Clamp_High_mask for underflow** Format: U1 If current frame is underflow and this mask is set, it would mask out MFX_VP8_Img_Status register. DW1.bit0. In another word, subsequent passes would be skipped
	5	**Final Bistream Buffer Overrun Enable Mask** Format: U1 Enable Final Bitstream Buffer Overrun detection feature.
	4	**Intermediate Bit Buffer Overrun Enable Mask** Format: U1 Enable Intermediate Bit Buffer Overrun detection feature.
	3	**Max Intra MB Bit Count Check Enable Mask**

MFX_VP8_Encoder_CFG

		Format: U1 Enable Max. Intra MB bit count check in Streamout.
	2	**Max Inter MB Bit Count Check Enable Mask** Format: U1 Enable Max. Inter MB bit count check in Streamout.
	1	**Min Frame Bit Count Rate Control Enable Mask** Format: U1 Enable Min. Frame Rate Control. This is a mask bit controlling if the condition of frame level bit count is less than or equal to FrameBitRateMin. **Value** / **Name** / **Description** 1 / / If (Total Frame Level Bit Counter) =< (Frame Bit Rate Minimum limit) Set bit[0] and bit[1] of MFX_VP8_IMAGE_STATUS Control Register. 0 / / Do not update bit[0] of MFX_VP8_IMAGE_STATUS Control Register.
	0	**Max Frame bit count Rate Control Enable Mask** Format: U1 Enable Max. Frame Rate Control. This is a mask bit controlling if the condition of frame level bit count is greater than or equal to FrameBitRateMax. **Value** / **Name** / **Description** 1 / / If (Total Frame Level Bit Counter) >= (Frame Bit Rate Maximum Limit) Set bit[0] and bit[1] of MFX_VP8_IMAGE_STATUS control register. 0 / / Do not update bit[0] of MFX_VP8_IMAGE_STATUS control register.
3	31:28	**Reserved** Format: MBZ
	27:16	**Max Intra MB Bit Count Limit** Format: U12 12-bit bit count for Max Intra MB Limit.
	15:12	**Reserved** Format: MBZ
	11:0	**Max Inter MB bit count** Format: U12 12-bit bit count for Max Inter MB Limit.
4	31	**Frame Bitrate Min Unit Mode**

MFX_VP8_Encoder_CFG

		Format: U1 This field is the Frame Bitrate Minimum Limit Units. **Value** / **Name** / **Description** 0h / Compatibility Mode / Frame BitRate Min Unit is in old mode **(128b/16Kb)** 1h / New Mode / Frame BitRate Min Unit is in new mode **(32byte/4Kb)**
	30	**Frame Bit Rate Min Unit** Format: U1 *This field is Frame Bitrate Minimum Mode.* **Value** / **Name** 0 / 32-B 1 / 4-KB
	29:16	**Frame Bit Rate Min** Format: U14 If either BRC Underflow or overflow is enabled. Frame Bit Rate Min and Frame Bit Rate Max need to be programmed with unambiguous values
	15	**Frame Bitrate Max Unit Mode** Format: U1 This field is the Frame Bitrate Maximum Limit Units. **Value** / **Name** / **Description** 0h / Compatibility Mode / Frame BitRate Max Unit is in old mode **(128b/16Kb)** 1h / New Mode / Frame BitRate Max Unit is in new mode **(32byte/4Kb)**
	14	**Frame Bit Rate Max Unit** Format: U1 *This field is Frame Bitrate Maximum Mode* **Value** / **Name** 0 / 32-B 1 / 4-KB
	13:0	**Frame Bit Rate Max** Format: U14 If either BRC Underflow or overflow is enabled. Frame Bit Rate Min and Frame Bit Rate Max need to be programmed with unambiguous values
5	31:24	**Frame Delta QIndex Max[3]**

MFX_VP8_Encoder_CFG		
		This field is the Frame level delta Qindex for total bit-count above FrameBitRateMax - First 1/8 Region. This field is used to calculate the suggested Frame Qindex into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame exceeds FrameBitRateMax but is within 1/8 of FrameBitRateMaxDelta above **FrameBitRateMax**; i.e., In the range of (FrameBitRateMax, (FrameBitRateMax + FrameBitRateMaxDelta » 3)].
	23:16	**Frame DeltaQ Index Max[2]** This field is the Frame level delta Qindex for bit-count above FrameBitRateMax - Above 1/8 and Below 1/4. This field is used to calculate the suggested Frame Qindex into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is between 1/8 and 1/4 of FrameBitRateMaxDelta above **FrameBitRateMax**; i.e., In the range of ((FrameBitRateMax + FrameBitRateMaxDelta » 3), (FrameBitRateMax+ FrameBitRateMaxDelta » 2)].
	15:8	**Frame Delta QIndex Max[1]** This field is the Frame level delta QINDEX for bit-count above FrameBitRateMax - Above 1/4 and Below 1/2. This field is used to calculate the suggested Frame QINDEX into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is between 1/4 and 1/2 of FrameBitRateMaxDelta above **FrameBitRateMax**; i.e., In the range of [(FrameBitRateMax+ FrameBitRateMaxDelta » 2), (FrameBitRateMax+ FrameBitRateMaxDelta » 1)].
	7:0	**Frame Delta QIndex Max [0]** This field is the Frame level delta QINDEX for bit-count above FrameBitRateMax - Above 1/2. This field is used to calculate the suggested Frame QINDEX into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is above FrameBitRateMax by more than half the distance of **FrameBitRateMax**; i.e., In the range of [(FrameBitRateMax + FrameBitRateMaxDelta » 1), ∞ (Infinite)].
6	31:24	**Frame Delta QIndex Min[3]** This field is the Frame level delta QINDEX for total bit-count below FrameBitRateMin - First 1/8 Region. This field is used to calculate the suggested Frame QINDEX into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is less than FrameBitRateMin and greater than or equal to 1/8 the distance of FrameBitRateMinDelta from **FrameBitRateMin**; i.e., In the range of [(FrameBitRateMin - FrameBitRateMinDelta » 3), FrameBitRateMin].
	23:16	**Frame Delta QIndex Min[2]** This field is the Frame level delta QINDEX for bit-count below FrameBitRateMin - Below 1/ 8 and Above 1/4. This field is used to calculate the suggested Frame QINDEX into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire

		frame is between one-eighth and quarter the distance of FrameBitRateMinDelta from **FrameBitRateMin**; i.e., In the range of [(FrameBitRateMin - FrameBitRateMinDelta » 2), (FrameBitRateMin- FrameBitRateMinDelta » 3)].
	15:8	**Frame Delta QIndex Min[1]** This field is the Frame level delta QINDEX for bit-count below FrameBitRateMin-Below 1/4 and Above 1/2. This field is used to calculate the suggested Frame QINDEX into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is between quarter and half the distance of FrameBitRateMinDelta from **FrameBitRateMin**; i.e., In the range of [(FrameBitRateMin - FrameBitRateMinDelta » 1), (FrameBitRateMin - FrameBitRateMinDelta » 2)].
	7:0	**Frame Delta QIndex Min[0]** This field is the Frame Level Delta QINDEX for bit-count below FrameBitRateMin - Below 1/2. This field is used to calculate the suggested Frame QINDEX into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is below FrameBitRateMin by more than half the distance of **FrameBitRateMin**; i.e., In the range of [0, (FrameBitRateMin - FrameBitRateMinDelta » 1).
7	31:0	**Per Segment Frame Delta QIndex Max[1]**
8	31:0	**Per Segment Frame Delta QIndex Min[1]**
9	31:0	**Per Segment Frame Delta QIndex Max[2]**
10	31:0	**Per Segment Frame Delta QIndex Min[2]**
11	31:0	**Per Segment Frame Delta QIndex Max[3]**
12	31:0	**Per Segment Frame Delta QIndex Min[3]**
13	31:24	**Frame Delta Loop Filter Max[3]** Format: U8 This field is the Frame level delta LoopFilter for total bit-count above FrameBitRateMax - First 1/8 region. This field is used to calculate the suggested Frame LoopFilter into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame exceeds FrameBitRateMax but is within 1/8 of FrameBitRateMaxDelta above FrameBitRateMax. i.e., in the range of (FrameBitRateMax, (FrameBitRateMax+ FrameBitRateMaxDelta » 3)].

MFX_VP8_Encoder_CFG

	23:16	**Frame Delta Loop Filter Max[2]** Format: U8 This field is the Frame level delta LoopFilter for bit-count above FrameBitRateMax - Above 1/8 and Below 1/4. This field is used to calculate the suggested Frame LoopFilter into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is between 1/8 and 1/4 of FrameBitRateMaxDelta above FrameBitRateMax. i.e., in the range of ((FrameBitRateMax + FrameBitRateMaxDelta » 3) and (FrameBitRateMax + FrameBitRateMaxDelta » 2)].
	15:8	**Fram eDelta Loop Filter Max[1]** Format: U8 This field is the Frame level delta LOOPFILTER for bit-count above FrameBitRateMax - Above1/ 4 and Below 1/2. This field is used to calculate the suggested Frame LOOPFILTER into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is between 1/4 and1/2 of FrameBitRateMaxDelta above FrameBitRateMax. i.e., in the range of ((FrameBitRateMax+ FrameBitRateMaxDelta » 2) and (FrameBitRateMax+ FrameBitRateMaxDelta » 1)].
	7:0	**Frame Delta Loop Filter Max[0]** Format: U8 This field is the Frame level delta LOOPFILTER for bit-count above FrameBitRateMax - Above 1/2. This field is used to calculate the suggested Frame LOOPFILTER into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is above FrameBitRateMax by more than half the distance of FrameBitRateMaxDelta. i.e., in the range of ((FrameBitRateMax+ FrameBitRateMaxDelta » 1), infinite).
14	31:24	**Frame Delta Loop Filter Min[3]** Format: U8 This field is the Frame level delta LOOPFILTER for total bit-count below FrameBitRateMin - First 1/8 region. This field is used to calculate the suggested Frame LOOPFILTER into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is less than FrameBitRateMin and greater than or equal to 1/8 the distance of FrameBitRateMinDelta from FrameBitRateMin. i.e., in the range of [(FrameBitRateMin - FrameBitRateMinDelta » 3), FrameBitRateMin).
	23:16	**Frame Delta Loop Filter Min[2]** Format: U8 This field is the Frame level delta LOOPFILTER for bit-count below

		FrameBitRateMin - Below 1/ 8 and Above 1/4. This field is used to calculate the suggested Frame LOOPFILTER into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is between one-eighth and quarter the distance of FrameBitRateMinDelta from FrameBitRateMin. i.e., in the range of [(FrameBitRateMin - FrameBitRateMinDelta » 2), (FrameBitRateMin - FrameBitRateMinDelta » 3)).
	15:8	**Frame Delta Loop Filter Min[1]** Format: U8 This field is the Frame level delta LOOPFILTER for bit-count below FrameBitRateMin- Below 1/4 and Above 1/2. This field is used to calculate the suggested Frame LOOPFILTER into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is between quarter and half the distance of FrameBitRateMinDelta from FrameBitRateMin. i.e., in the range of [(FrameBitRateMin - FrameBitRateMinDelta » 1) and (FrameBitRateMin - FrameBitRateMinDelta » 2)).
	7:0	**Frame Delta Loop Filter Min[0]** Format: U8 This field is the Frame Level Delta LOOPFILTER for bit-count below FrameBitRateMin - Below 1/ 2. This field is used to calculate the suggested Frame LOOPFILTER into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is below FrameBitRateMin by more than half the distance of FrameBitRateMinDelta. i.e., in the range of [0, (FrameBitRateMin - FrameBitRateMinDelta » 1).
15	31:0	**Per Segment Frame Delta LoopFilter Max[1]**
16	31:0	**Per Segment Frame Delta LoopFilter Min[1]**
17	31:0	**Per Segment Frame Delta LoopFilter Max[2]**
18	31:0	**Per Segment Frame Delta LoopFilter Min[2]**
19	31:0	**Per Segment Frame Delta LoopFilter Max[3]**
20	31:0	**Per Segment Frame Delta LoopFilter Min[3]**
21	31	**Reserved**

MFX_VP8_Encoder_CFG

	30:16	**FrameBitRateMinDelta** Format: U15 This field is used to select the frame delta QINDEX when FrameBitRateMin Is exceeded. It shares the same FrameBitrateMinUnit. **Value** / **Name** / **Description** [0-4095] / / When FrameBitrateMinUnit is in Bytes, this range is in Bytes. When FrameBitrateMinUnit is in KB, this range is in KB units.
	15	**Reserved**
	14:0	**Frame Bit Rate Max Delta** Format: U15 This field is used to select the frame delta QINDEX when FrameBitRateMax Is exceeded. It shares the same FrameBitrateMaxUnit. **Value** / **Name** / **Description** [0-4095] / / When FrameBitrateMinUnit is in Bytes, this range is in Bytes. When FrameBitrateMinUnit is in KB, this range is in KB units.
22	31:24	**Reserved** Format: MBZ
	23	**Show Frame** Format: U1 VP8 Frame Tag, Show Frame Field
	22:20	**Bitstream Format Version** Format: U3 VP8 Frame Tag, Verison Field
	19:18	**Reserved** Format: MBZ
	17:16	**Min Frame WSize Unit** Format: U2 **Value** / **Name** / **Description** 0h / Compatibility Mode / MinFrameWSizeUnit is in old mode (128b/16Kb) 1h / New Mode / MinFrameWSizeUnit is in new mode (32byte/4Kb)
	15:0	**Min Frame WSize** Exists If: //Encoder Only This field (in Word, 16-bit) is specified to compensate for Intel® Rate Control. Zero padding would be performed.

MFX_VP8_Encoder_CFG		
23	31:16	**Vertical_Size_Code** Format: U16 Frame Tag Vertical Size Code, composed of {VerticalScale[15:14], FrameHeight[13:0]}
	15:0	**Horizontal_Size_Code** Format: U16 Frame Tag Horizontal Size Code, composed of {HorizontalScale[15:14], FrameWidth[13:0]}
24	31:0	**Frame Header Bit Count** Format: U32 Binarized Header Bit Count.
25	31:0	**Frame Header Bin Buffer Qindex Update Pointer** Format: U32 Binarized Header Qindex Update Pointer If Segment Enabled and UpdateSegmentFeature enabled, 4 per segment Qindices would be updated in Binarized header (Only ABS mode supported). Else Base Qindex would be updated
26	31:0	**Frame Header Bin Buffer LoopFilter Update Pointer** Format: U32 Binarized Header LoopFilter Update Pointer If Segment Enabled and UpdateSegmentFeature enabled, 4 per segment LoopFilters would be updated in Binarized header (Only ABS mode supported). Else Base LoopFilter would be updated.
27	31:0	**Frame Header Bin Buffer Token Update Pointer** Format: U32 Binarized Header TokenUpdate Pointer
28	31:0	**Frame Header Bin Buffer MVUpdate Pointer** Format: U32 Binarized Header MVUpdate Pointer.
29 **Programming Notes:** The only value permitted for CV7	31:28	**ClampValues - CV7**
	27:24	**CV6**
	23:20	**CV5**
	19:16	**CV4**

MFX_VP8_Encoder_CFG

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

Format:	U4

If the magnitude of coefficients at locations assigned with CV0 (mapping shown below) exceeds 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 4x4 frame block, each coefficient is mapped to one of the eight CV values as following:

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

For 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

Value	Name
0-15	

MFX_VP8_PAK_OBJECT

MFX_VP8_PAK_OBJECT

Source: VideoCS

Length Bias: 2

The MFX_VP8_PAK_OBJECT command is the second primitive command for the VP8 Encoding Pipeline. The MV Data portion of the bitstream is loaded as indirect data object. Before issuing a MFX_VP8_PAK_OBJECT command, all VP8 MFX states need to be valid; therefore the commands used to set these states need to have been issued prior to the issue of this command. MB record must be consecutive with no gaps, hence we do not need MB(x,y) in each MB command. Internal counter will keep track of the current MB address, starting from the first MB. MFX_VP8_PAK_OBJECT command follows the MbType definition like MFD. Encoding statistical data such as the total size of the output bitstream are provided through MMIO registers. Software may access these registers through MI_STORE_REGISTER_MEM command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_VP8_PAK_OBJECT Format: OpCode
	26:24	**Media Command Opcode** Default Value: 4h VP8_ENC Format: OpCode
	23:21	**SubOpcode A** Default Value: 2h Format: OpCode
	20:16	**SubOpcode B** Default Value: 9h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 5h DWORD_COUNT_n Format: =n Length -2

MFX_VP8_PAK_OBJECT

1	31:30	**Reserved** Format: MBZ
	29	**Enable Inline MV data** Format: Enable This field denotes if the MV data will be sent inline following the other inline data instead of being indirect.
	28:10	**Reserved** Format: MBZ
	9:0	**Indirect PAK-MV Data Length** Format: U10 This field provides the length in bytes of the indirect data, which contains all the MVs for the current MB (in any partitioning and subpartitioning form). A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect PAK-MV Data Start Address field is ignored. This field must have the same alignment as the Indirect PAK-MV Data Start Address. This field must be DW aligned (since each MV is 4 bytes in size).Driver has to derived this field from MVsize (MVquantity in DXVA, exact size) *4 bytes per MV.
2	31:29	**Reserved** Format: MBZ
	28:0	**Indirect PAK-MV Data Start Address Offset** This field specifies the memory starting address (offset) of the MV data to be fetched into PAK Subsystem for processing. This pointer is relative to the MFC Indirect PAK-MV Object Base Address. Hardware ignores this field if indirect data is not present, i.e. the Indirect PAK-MV Data Length is set to 0. It is a Dword aligned address in all AVC encoding configuration, since each MV is 4 bytes in size. Value: [0,512MB) — Name:
3..6	127:0	**Inline Data** All the required MB level controls and parameters for encoding are captured as Inline Data Description - VP8 PAK OBJECT. It has a fixed size of 4 DWs. Its definition is described in the next section.

MFX_VP8_PIC_STATE

MFX_VP8_PIC_STATE

Source: VideoCS

Length Bias: 2

This must be the very first command to issue after the surface state, the pipe select and base address setting commands and must be issued before MFX_VP8_IMG_STATE.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Video Codec Format: OpCode
	26:24	**Media Command OpCode** Default Value: 4h VP8 Format: OpCode
	23:21	**Sub Opcode A** Default Value: 0h VP8 Common Format: OpCode
	20:16	**Sub Opcode B** Default Value: 0h MFX_VP8_PIC_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Value / Name / Description 000h / Excludes DWord (0,1) **[Default]** / A special case to provide a dummy image state for stitch mode operation. In this case, fields in DW1 which is part of the dummy image state command are ignored by hardware." 024h / / Used for normal decode and encode mode
1	31:24	**Reserved** Exists If: //Decoder / Encoder Format: MBZ

MFX_VP8_PIC_STATE

	23:16	**Frame Height Minus 1** Exists If: //Decoder / Encoder Format: U8 Picture Height in integer number of MBs minus 1, so the min pic height can be program is 16 rows of pixels.
	15:8	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	7:0	**Frame Width Minus 1** Exists If: //Decoder / Encoder Format: U8 Picture Width in integer number of MBs minus 1, so the min pic width can be program is 16 pixels.
2	31:26	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	25:24	**Log2 Num of Partition** Exists If: //Decoder / Encoder Format: U2

Value	Name
0	1 Token partition
1	2 Token partition
2	4 Token partition
3	8 Token partition

	23:19	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	18:16	**Deblock Sharpness Level** Exists If: //Decoder / Encoder Format: U3 Specify the sharpness level, as one of the regular deblocking strength control parameters.

Programming Notes
Set to 0 to disable the use of sharpness control.

	15:14	**Reserved**

Exists If:	//Decoder / Encoder
Format:	MBZ

13 **Alternate Ref Pic MV SignBias Flag**

Exists If:	//Decoder / Encoder

Alternate Reference Picture MV sign bias flag, specified for non-key frame only.

12 **Golden Ref Picture MV SignBias Flag**

Exists If:	//Decoder / Encoder

Golden Reference Picture MV sign bias flag, specified for non-key frame only.

11 **Mode Reference Loop Filter Delta Enabled**

Exists If:	//Decoder / Encoder

Value	Name	Description
0		Mode or Reference Loop Filter Delta Adjustment for current frame is disabled.
1		Mode or Reference Loop Filter Delta Adjustment for current frame is enabled.

10 **MB NoCoeff SkipFlag**

Exists If:	//Decoder / Encoder

Frame level control if Skip MB (with no non-zero coefficient) is allowed or not.

Value	Name	Description
0		All MBs will have its MB level signaling mb_skip_coeff forced to 0. That is, no skip of coefficient record in the bitstream (even their values are all 0s)
1		Skip MB is enabled in the per MB record.

9 **Update MBSegment Map Flag**

Exists If:	//Decoder / Encoder

Value	Name	Description
0		Disable segmentation update
1		Enable segmentation update, and to enable reading segment_id for each MB.

8 **Segment Enable Flag**

Exists If:	//Decoder / Encoder

Value	Name	Description
0		Disable Segmentation processing in the current frame
1		Enable Segmentation processing in the current frame

7 **Segmentation ID StreamIn Enable**

Exists If:	//Decoder Only

Value	Name
0	StreamIn Disabled
1	StreamIn Enabled

Programming Notes
When 0, no input needed.

7:6 **Reserved**

Exists If:	//Encoder Only
Format:	MBZ

6 **Segmentation ID StreamOut Enable**

Exists If:	//Decoder Only

Value	Name
0	StreamOut Disabled
1	StreamOut Enabled

Programming Notes
When 0, no output needed.

5 **sKeyFrameFlag**

Exists If:	//Decoder / Encoder

Value	Name
0	Non-Key Frame (P-Frame)
1	Key Frame (I-Frame)

4 **DBLKFilterType**

Exists If:	//Decoder / Encoder

To specify VP8 Profile of operation.

Value	Name	Description
0		Use a full feature normal deblocking filter
1		Use a simple filter for deblocking

3:2 **Reserved**

Exists If:	//Decoder / Encoder
Format:	MBZ

1 **Chroma Full Pixel MC Filter Mode**

Exists If:	//Decoder / Encoder

To specify VP8 Profile of operation.

MFX_VP8_PIC_STATE

Value	Name	Description
0		Chroma MC filter operates in sub-pixel mode
1		Chroma MC filter only operates in full pixel position, i.e. no sub-pixel interpolation.

0 — MC Filter Select

Exists If:	//Decoder / Encoder

To specify VP8 Profile of operation.

Value	Name	Description
0		6-tap filter (regular filter mode)
1		2-tap bilinear filter (simple profile/version mode)

3 — 31:30 Reserved

Exists If:	//Decoder / Encoder
Format:	MBZ

29:24 — DBLKFilterLevel for Segment3

Exists If:	//Decoder / Encoder
Format:	U6

Value	Name	Description
0	Signifies disable in loop deblocking operation	This is used to set a VP8 profile without in loop deblocker.

Programming Notes
There are max 4 segments per frame, each segment can have its own deblocking filter level. When segmentation is disabled, only segment 0 parameter is used for the entire frame.

23:22 — Reserved

Exists If:	//Decoder / Encoder
Format:	MBZ

21:16 — DBLKFilterLevel for Segment2

Exists If:	//Decoder / Encoder
Format:	U6

Value	Name	Description
0	Signifies disable in loop deblocking operation	This is used to set a VP8 profile without in loop deblocker.

Programming Notes

		There are max 4 segments per frame, each segment can have its own deblocking filter level. When segmentation is disabled, only segment 0 parameter is used for the entire frame.
	15:14	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	13:8	**DBLKFilterLevel for Segment1** Exists If: //Decoder / Encoder Format: U6 Value 0 – Name: Signifies disable in loop deblocking operation – Description: This is used to set a VP8 profile without in loop deblocker. **Programming Notes** There are max 4 segments per frame, each segment can have its own deblocking filter level. When segmentation is disabled, only segment 0 parameter is used for the entire frame.
	7:6	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	5:0	**DBLKFilterLevel for Segment0** Exists If: //Decoder / Encoder Format: U6 Value 0 – Name: Signifies disable in loop deblocking operation – Description: This is used to set a VP8 profile without in loop deblocker. **Programming Notes** There are max 4 segments per frame, each segment can have its own deblocking filter level. When segmentation is disabled, only segment 0 parameter is used for the entire frame.
4	31	**Reserved** Exists If: //Encoder Only Format: MBZ
	31:0	**Reserved** Exists If: //Decoder Only

	Format:	MBZ
30:24	**Seg 3 Qindex**	
	Exists If:	//Encoder Only
	Format:	U7
	Quantizer Value for Segment ID 3	
23	**Reserved**	
	Exists If:	//Encoder Only
	Format:	MBZ
22:16	**Seg 2 Qindex**	
	Exists If:	//Encoder Only
	Format:	U7
	Quantizer Value for Segment ID 2	
15	**Reserved**	
	Exists If:	//Encoder Only
	Format:	MBZ
14:8	**Seg 1 Qindex**	
	Exists If:	//Encoder Only
	Format:	U7
	Quantizer Value for Segment ID 1	
7	**Reserved**	
	Exists If:	//Encoder Only
	Format:	MBZ
6:0	**Seg 0 Qindex**	
	Exists If:	//Encoder Only
	Format:	U7
	Quantizer Value for Segment ID 0.	
	Programming Notes	

		This is the **[Default]** Qindex
5	31:29	**Reserved** Exists If: //Encoder Only Format: MBZ
	31:0	**Reserved** Exists If: //Decoder Only Format: MBZ
	28	**UVac Qindex Delta Sign** Exists If: //Encoder Only Format: U1 Sign of Quantization index delta for UVac
	27:24	**UVac QindexDelta** Exists If: //Encoder Only Format: U4 Absolute Quantization index delta for UVac
	23:21	**Reserved** Exists If: //Encoder Only Format: MBZ
	20	**UVdc Qindex Delta Sign** Exists If: //Encoder Only Format: U1 Sign of Quantization index delta for UVdc
	19:16	**UVdc Qindex Delta** Exists If: //Encoder Only Format: U4 Absolute Quantization index delta for UVdc

	15:13	**Reserved** Exists If: //Encoder Only Format: MBZ
	12	**Y2ac Qindex Sign** Exists If: //Encoder Only Format: U1 Sign of Quantization index delta for Y2ac
	11:8	**Y2ac Qindex Delta** Exists If: //Encoder Only Format: U4 Absolute Quantization index delta for Y2ac
	7:5	**Reserved** Exists If: //Encoder Only Format: MBZ
	4	**Y2ac Qindex Delta Sign** Exists If: //Encoder Only Format: U1 Sign of Quantization index delta for Y2dc This is the **[Default]** Qindex Delta Sign
	3:0	**Y2dc Qindex Delta** Exists If: //Encoder Only Format: U4 Absolute Quantization index delta for Y2dc This is the **[Default]** Qindex Delta
6	31:5	**Reserved** Exists If: //Encoder Only

		Format: MBZ
	31:0	**Reserved** Exists If: //Decoder Only Format: MBZ
	4	**Y1dc Qindex Delta Sign** Exists If: //Encoder Only Format: U1 Sign of Quantization index delta for Y1dc This is the **[Default]**Qindex Delta Sign
	3:0	**Y1dc Qindex Delta** Exists If: //Encoder Only Absolute Quantization index delta for Y1dc This is the **[Default]**Qindex Delta
7	31:15	**Reserved** Exists If: //Encoder Only Format: MBZ
	31:0	**Reserved** Exists If: //Decoder Only Format: MBZ
	14:8	**Clamp Qindex high** Exists If: //Encoder Only Format: U7 Maximum Clamp Value for Qindex used in quantization.
	7	**Reserved** Exists If: //Encoder Only Format: MBZ

MFX_VP8_PIC_STATE

	6:0	**Clamp Qindex Low** Exists If: //Encoder Only Format: U7 Minimum Clamp Value for Qindex used in quantization.
8	31:25	**Reserved** Exists If: //Decoder Only Format: MBZ
	31:0	**Reserved** Exists If: //Encoder Only Format: MBZ
	24:16	**Quantizer Value [1][BlockType3=UVAC]** Exists If: //Decoder Only Format: U9 **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
	15:9	**Reserved** Exists If: //Decoder Only Format: MBZ
	8:0	**Quantizer Value [1][BlockType2=UVDC]** Exists If: //Decoder Only **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
9	31:25	**Reserved** Exists If: //Decoder Only Format: MBZ
	31:0	**Reserved** Exists If: //Encoder Only Format: MBZ
	24:16	**Quantizer Value [1][BlockType5=Y2AC]** Exists If: //Decoder Only Format: U9 **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
	15:9	**Reserved** Exists If: //Decoder Only

MFX_VP8_PIC_STATE

		Format: MBZ
	8:0	**Quantizer Value [1][BlockType4=Y2DC]** Exists If: //Decoder Only **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
10	31:25	**Reserved** Exists If: //Decoder Only Format: MBZ
	31:0	**Reserved** Exists If: //Encoder Only Format: MBZ
	24:16	**Quantizer Value [2][BlockType1=Y1AC]** Exists If: //Decoder Only Format: U9 **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
	15:9	**Reserved** Exists If: //Decoder Only Format: MBZ
	8:0	**Quantizer Value [2][BlockType0=Y1DC]** Exists If: //Decoder Only **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
11	31:25	**Reserved** Exists If: //Decoder Only Format: MBZ
	31:0	**Reserved** Exists If: //Encoder Only Format: MBZ
	24:16	**Quantizer Value [2][BlockType3=UVAC]** Exists If: //Decoder Only Format: U9 **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
	15:9	**Reserved** Exists If: //Decoder Only Format: MBZ

MFX_VP8_PIC_STATE

	8:0	**Quantizer Value [2][BlockType2=UVDC]** Exists If: //Decoder Only **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
12	31:25	**Reserved** Exists If: //Decoder Only Format: MBZ
	31:0	**Reserved** Exists If: //Encoder Only Format: MBZ
	24:16	**Quantizer Value [2][BlockType5=Y2AC]** Exists If: //Decoder Only Format: U9 **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
	15:9	**Reserved** Exists If: //Decoder Only Format: MBZ
	8:0	**Quantizer Value [2][BlockType4=Y2DC]** Exists If: //Decoder Only **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
13	31:25	**Reserved** Exists If: //Decoder Only Format: MBZ
	31:0	**Reserved** Exists If: //Encoder Only Format: MBZ
	24:16	**Quantizer Value [3][BlockType1=Y1AC]** Exists If: //Decoder Only Format: U9 **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
	15:9	**Reserved** Exists If: //Decoder Only Format: MBZ
	8:0	**Quantizer Value [3][BlockType0=Y1DC]**

MFX_VP8_PIC_STATE

		Exists If: //Decoder Only **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
14	31:25	**Reserved** Exists If: //Decoder Only Format: MBZ
	31:0	**Reserved** Exists If: //Encoder Only Format: MBZ
	24:16	**Quantizer Value [3][BlockType3=UVAC]** Exists If: //Decoder Only Format: U9 **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
	15:9	**Reserved** Exists If: //Decoder Only Format: MBZ
	8:0	**Quantizer Value [3][BlockType2=UVDC]** Exists If: //Decoder Only **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
15	31:25	**Reserved** Exists If: //Decoder Only Format: MBZ
	31:0	**Reserved** Exists If: //Encoder Only Format: MBZ
	24:16	**Quantizer Value [3][BlockType5=Y2AC]** Exists If: //Decoder Only Format: U9 **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
	15:9	**Reserved** Exists If: //Decoder Only Format: MBZ
	8:0	**Quantizer Value [3][BlockType4=Y2DC]** Exists If: //Decoder Only

		Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]
16	31:6	**CoeffProbability StreamIn Base Address** Exists If: //Decoder Only Format: StreamInAddress[31:6] 64 bytes aligned buffer in linear format. (not tile for better performance) It is specified for non-key frame only. It is the final computed probability table for parsing Coeff in the bitstream. The buffer is unsigned 8-bit * 1056 entries (CoeffProbs[4][8][3][11].
	31:0	**Reserved** Exists If: //Encoder Only Format: MBZ
	5:0	**Reserved** Exists If: //Decoder Only Format: MBZ
17	31:16	**Reserved** Exists If: //Decoder Only Format: MBZ
	31:0	**Reserved** Exists If: //Encoder Only Format: MBZ
	15:0	**CoeffProbability StreamIn Address** Exists If: //Decoder Only This field is for the upper range of CoeffProbability StreamIn Address
18	31:15	**Reserved** Exists If: //Decoder Only Format: MBZ
	31:7	**Reserved** Exists If: //Encoder Only

Format:	MBZ

14:13 **CoeffProbability StreamIn - Tiled Resource Mode**

Exists If:	//Decoder Only
Format:	U2

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

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

12:11 **Reserved**

Exists If:	//Decoder Only
Format:	MBZ

10 **CoeffProbability StreamIn - Memory Compression Mode**

Exists If:	//Decoder Only
Format:	U1

Distinguishes Vertical from Horizontal compression. Please refer to vol1a **Memory Data Formats** chapter, **Media Memory Compression** section for more details.

Value	Name
0	Horizontal Compression Mode
1	Vertical Compression Mode

9 **CoeffProbability StreamIn - Memory Compression Enable**

Exists If:	//Decoder Only
Format:	Enable

Memory compression will be attempted for this surface.

8:7 **CoeffProbability StreamIn - Arbitration Priority Control**

Exists If:	//Decoder Only
Format:	U2

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

Value	Name

		00b: Highest priority 01b: Second highest priority 10b: Third highest priority 11b: Lowest priority
	6:1	**CoeffProbability StreamIn Address - Index to Memory Object Control State (MOCS) Tables** Exists If: //Encoder Only Format: U6 The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.
	0	**Reserved**
19	31:24	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	23:16	**MBSegmentIDTreeProbs[2]** Exists If: //Decoder / Encoder Format: U8 MBSegmentIDTreeProbs[2:0] probability tree table for CPBAC parsing Segment_ID of each MB.
	15:8	**MBSegmentIDTreeProbs[1]** Exists If: //Decoder / Encoder Format: U8 MBSegmentIDTreeProbs[2:0] probability tree table for CPBAC parsing Segment_ID of each MB.
	7:0	**MBSegmentIDTreeProbs[0]** Exists If: //Decoder / Encoder Format: U8 MBSegmentIDTreeProbs[2:0] probability tree table for CPBAC parsing Segment_ID of each MB.
20	31:24	**MBNoCoeffSkipFalseProb** Exists If: //Decoder / Encoder Format: U8 8-bit probability value for CPBAC parsing of the MBNoCoeffSkip Flag in the bistream.

MFX_VP8_PIC_STATE

	23:16	**IntraMBProb** Exists If: //Decoder / Encoder Format: U8 8-bit probability value for CPBAC parsing of the intra or inter MB type flag in the bitstream.
	15:8	**InterPredFromLastRefProb** Exists If: //Decoder / Encoder Format: U8 8-bit probability value for CPBAC parsing of the flag in the bitstream that determines which reference frame to be used for the current MB motion compensation.
	7:0	**InterPredFromGRefRefProb** Exists If: //Decoder / Encoder Format: U8 8-bit probability value for CPBAC parsing of the flag in the bitstream that determines which reference frame to be used for the current MB motion compensation.
21	31:24	**YModeProb[3]** Exists If: //Decoder / Encoder Format: U8 YModeProb[3:0] probability tree table for CPBAC parsing Luma MBType of each MB.
	23:16	**YModeProb[2]** Exists If: //Decoder / Encoder Format: U8 YModeProb[3:0] probability tree table for CPBAC parsing Luma MBType of each MB.
	15:8	**YModeProb[1]** Exists If: //Decoder / Encoder Format: U8 YModeProb[3:0] probability tree table for CPBAC parsing Luma MBType of each MB.
	7:0	**YModeProb[0]** Exists If: //Decoder / Encoder Format: U8 YModeProb[3:0] probability tree table for CPBAC parsing Luma MBType of each MB.
22	31:24	**Reserved** Exists If: //Decoder / Encoder

		Format: MBZ
	23:16	**UVModeProb[2]** Exists If: //Decoder / Encoder Format: U8 UVModeProb[2:0] probability tree table for CPBAC parsing Chroma MBType of each MB.
	15:8	**UVModeProb[1]** Exists If: //Decoder / Encoder Format: U8 UVModeProb[2:0] probability tree table for CPBAC parsing Chroma MBType of each MB.
	7:0	**UVModeProb[0]** Exists If: //Decoder / Encoder Format: U8 UVModeProb[2:0] probability tree table for CPBAC parsing Chroma MBType of each MB.
23	31:24	**MVUpdateProbs[0][3]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	23:16	**MVUpdateProbs[0][2]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	15:8	**MVUpdateProbs[0][1]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	7:0	**MVUpdateProbs[0][0]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].

MFX_VP8_PIC_STATE

24	31:24	**MVUpdateProbs[0][7]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	23:16	**MVUpdateProbs[0][6]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	15:8	**MVUpdateProbs[0][5]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	7:0	**MVUpdateProbs[0][4]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
25	31:24	**MVUpdateProbs[0][11]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB.
	23:16	**MVUpdateProbs[0][10]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB.
	15:8	**MVUpdateProbs[0][9]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB.

MFX_VP8_PIC_STATE

	7:0	**MVUpdateProbs[0][8]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB.
26	31:24	**MVUpdateProbs[0][15]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	23:16	**MVUpdateProbs[0][14]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	15:8	**MVUpdateProbs[0][13]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	7:0	**MVUpdateProbs[0][12]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
27	31:24	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	23:16	**MVUpdateProbs[0][18]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].

MFX_VP8_PIC_STATE

	15:8	**MVUpdateProbs[0][17]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	7:0	**MVUpdateProbs[0][16]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
28	31:24	**MVUpdateProbs[1][3]** Exists If: //Decoder Only Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	23:16	**MVUpdateProbs[1][2]** Exists If: //Decoder Only Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	15:8	**MVUpdateProbs[1][1]** Exists If: //Decoder Only Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	7:0	**MVUpdateProbs[1][0]** Exists If: //Decoder Only Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
29	31:24	**MVUpdateProbs[1][7]** Exists If: //Decoder / Encoder Format: U8

		MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	23:16	**MVUpdateProbs[1][6]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	15:8	**MVUpdateProbs[1][5]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	7:0	**MVUpdateProbs[1][4]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
30	31:24	**MVUpdateProbs[1][11]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	23:16	**MVUpdateProbs[1][10]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	15:8	**MVUpdateProbs[1][9]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	7:0	**MVUpdateProbs[1][8]**

MFX_VP8_PIC_STATE

		Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
31	31:24	**MVUpdateProbs[1][15]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	23:16	**MVUpdateProbs[1][14]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	15:8	**MVUpdateProbs[1][13]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	7:0	**MVUpdateProbs[1][12]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
32	31:24	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	23:16	**MVUpdateProbs[1][18]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	15:8	**MVUpdateProbs[1][17]**

		Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	7:0	**MVUpdateProbs[1][16]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
33	31	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	30:24	**RefLFDelta3 (for ALTREF FRAME)** Exists If: //Decoder / Encoder Format: S6 2's Compliment Delta value for reference frame based adjustment of the MB-level's filter level value. RefLFDeltas [ref_frametype = 0 to 3] **Programming Notes** Please note that although RefDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.
	23	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	22:16	**RefLFDelta2 (for GOLDEN FRAME)** Exists If: //Decoder / Encoder Format: S6 2's Compliment Delta value for reference frame based adjustment of the MB-level's filter level value. RefLFDeltas [ref_frametype = 0 to 3] **Programming Notes** Please note that although RefDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.
	15	**Reserved**

		Exists If: //Decoder / Encoder Format: MBZ
	14:8	**RefLFDelta1 (for LAST FRAME)** Exists If: //Decoder / Encoder Format: S6 2's Compliment Delta value for reference frame based adjustment of the MB-level's filter level value. RefLFDeltas [ref_frametype = 0 to 3] **Programming Notes** Please note that although RefDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.
	7	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	6:0	**RefLFDelta0 (for INTRA FRAME)** Exists If: //Decoder / Encoder Format: S6 2's Compliment Delta value for reference frame based adjustment of the MB-level's filter level value. RefLFDeltas [ref_frametype = 0 to 3] **Programming Notes** Please note that although RefDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.
34	31	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
	30:24	**ModeLFDelta3 (for SPLITMV mode)** Exists If: //Decoder / Encoder Format: S6 2's Compliment Delta value for mode based adjustment of the MB-level's filter level value. ModeLFDeltas[MB_Type = 0 to 3] **Programming Notes** Please note that although ModeLFDelta is signed 2's complement, bitstream is sign bit + 6 bit

	magnitude.
23	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
22:16	**ModeLFDelta2 (for Nearest, Near and New mode)** Exists If: //Decoder / Encoder Format: S6 2's Compliment Delta value for mode based adjustment of the MB-level's filter level value. ModeLFDeltas[MB_Type = 0 to 3] **Programming Notes** Please note that although ModeLFDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.
15	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
14:8	**ModeLFDelta1(for ZEROMV mode)** Exists If: //Decoder / Encoder Format: S6 2's Compliment Delta value for mode based adjustment of the MB-level's filter level value. ModeLFDeltas[MB_Type = 0 to 3] **Programming Notes** Please note that although ModeLFDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.
7	**Reserved** Exists If: //Decoder / Encoder Format: MBZ
6:0	**ModeLFDelta0 (for B_PRED mode)** Exists If: //Decoder / Encoder Format: S6 2's Compliment Delta value for mode based adjustment of the MB-level's filter level value. ModeLFDeltas[MB_Type = 0 to 3]

		Programming Notes Please note that although ModeLFDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.
35	31:0	**Segmentation ID Stream Base Address** Exists If: //Decoder Only Format: StreamAddress[31:0] 64 bytes linear aligned buffer It is specified when **SegmentationIDStreamInEnable** or **SegmentationIDStreamOutEnable** is specified. **Programming Notes** Each cache has only 8 bits for 4 segmentation ID from 4 continuous MBs.
36	31:16	**Reserved** Exists If: //Decoder Only Format: MBZ
	15:0	**Segmentation ID Stream Base Address [47:32]** Exists If: //Decoder Only This field is for the upper range of **Segmentation ID Stream Base Address**
37	31:15	**Reserved** Exists If: //Decoder Only Format: MBZ
	14:13	**Segmentation ID Stream - Tiled Resource Mode** Exists If: //Decoder Only Format: U2 **For Media Surfaces:** This field specifies the tiled resource mode. **Value** / **Name** / **Description** 0h / TRMODE_NONE / No tiled resource 1h / TRMODE_TILEYF / 4KB tiled resources 2h / TRMODE_TILEYS / 64KB tiled resources 3h / Reserved /
	12:11	**Reserved**

MFX_VP8_PIC_STATE

	Exists If:	//Decoder Only
	Format:	MBZ

10 Segmentation ID Stream - Memory Compression Mode

Format:	U1

Distinguishes Vertical from Horizontal compression. Please refer to vol1a **Memory Data Formats** chapter, **Media Memory Compression** section for more details.

Value	Name
0	Horizontal Compression Mode
1	Vertical Compression Mode

9 Segmentation ID Stream - Memory Compression Enable

Format:	Enable

Memory compression will be attempted for this surface.

8:7 Segmentation ID Stream - Arbitration Priority Control

Exists If:	//Decoder Only
Format:	U2

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

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

6:1 CoeffProbability StreamIn Address - Index to Memory Object Control State (MOCS) Tables

Format:	U6

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

0 Reserved

MFX_WAIT

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 pipeline. This 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**
	7:6	**Reserved** Format: MBZ
	5:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2

MI_ARB_CHECK

MI_ARB_CHECK

Source: VideoEnhancementCS

Length Bias: 1

Description

The command allows software to add preemption points in the ring buffer. The command streamer will preempt in the case arbitration is enabled, there is a pending execution list and this command is currently being parsed.

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_CHECK

MI_ARB_CHECK

Source: BlitterCS

Length Bias: 1

Description

The command allows software to add preemption points in the ring buffer. The command streamer will preempt in the case arbitration is enabled, there is a pending execution list and this command is currently being parsed.

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

MI_ARB_CHECK

Source: RenderCS

Length Bias: 1

Description

The command allows software to add preemption points in the ring buffer. The command streamer will preempt in the case arbitration is enabled, there is a pending execution list and this command is currently being parsed.

Programming Notes

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

MI_ARB_CHK command must not be programmed in INDIRECT_CTX and BB_PER_CTX_PTR buffers.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 05h MI_ARB_CHECK Format: OpCode
	22:0	**Reserved** Format: MBZ

MI_ARB_CHECK

MI_ARB_CHECK

Source: VideoCS

Length Bias: 1

Description

The command allows software to add preemption points in the ring buffer. The command streamer will preempt in the case arbitration is enabled, there is a pending execution list and this command is currently being parsed.

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

MI_ARB_ON_OFF

Source: CommandStreamer

Length Bias: 1

Description

The MI_ARB_ON_OFF instruction is used to disable/enable context switching. This instruction can be used to prevent submission of a new execlist from interrupting a command sequence, however lite restore preemption is allowed with in the arbitration disabled command execution zone. Note that context switching will remain disabled until re-enabled through use of this command. This command will also prevent a switch in the case of waiting on events, running out of commands. These will effectively hang the device if allowed to occur while arbitration is off (context switching is disabled.) This command should always be used as an off-on pair with the sequence of instructions to be protected from context switch between MI_ARB_OFF and MI_ARB_ON. Software must use this arbitration control with caution since it has the potential to increase the response time of the Render Engine to pre-emption requests. This is a privileged command; it will not be effective (will be converted to a no-op) if executed from within a non-privileged batch buffer.

The MI_ARB_ON_OFF instruction is used to disable/enable context switching. Context switching could be either due to preemption or unsuccessful wait for events or semaphore waits. This instruction can be used to prevent submission of a new execlist from interrupting a command sequence, however lite restore preemption is allowed with in the arbitration disabled command execution zone. Note that context switching will remain disabled until re-enabled through use of this command. This command will also prevent a switch in the case of waiting on events, running out of commands. These will effectively hang the device if allowed to occur while arbitration is off (context switching is disabled.)

Programming Notes

This command must be always be programmed in pairs of off/on in the same command dispatch. Sequence of instructions to be protected from context switch or preemption must be programmed between the MI_ARB_OFF and MI_ARB_ON. Software must use this arbitration control with caution since it has the potential to increase the response time of the Render Engine to pre-emption requests. This is a privileged command; it will not be effective (will be converted to a no-op) if executed from within a non-privileged batch buffer.

MI_ARB_ON_OFF command must not be programmed as part of the POSH command execution.

DWord	Bit	Description	
0	31:29	**Command Type**	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	**MI Command Opcode**	
		Default Value:	08h MI_ARB_ON_OFF
		Format:	OpCode
	22:2	**Reserved**	
		Format:	MBZ

MI_ARB_ON_OFF

	1	**Arbitration Mode** Source: RenderCS Format: Enable This bit controls whether or not lite restore is allowed when arbitration is disabled thru clearing the Arbitration Enable bit. If arbitration is enabled then the value of this bit does not change the behavior of the hardware. **Value** / **Name** 0h: Allow Lite Restore **[Default]** 1h: Lite Restore Disabled
	0	**Arbitration Enable** Format: Enable This field enables or disables context switches due to pre-emption (a new execlist). **Value** / **Name** 1: Arbitration Enabled **[Default]** 0: Arbitration Disabled

MI_ATOMIC

MI_ATOMIC

Source: BSpec

Length Bias: 2

Description

MI_ATOMIC is used to carry atomic operation on data in graphics memory. Atomic operations are supported on data granularity of 4B, 8B and 16B. The atomic operation leads to a read-modify-write operation on the data in graphics memory with the option of returning value. The data in graphics memory is modified by doing arithmetic and logical operation with the inline/indirect data provided with the MI_ATOMIC command. Inline/Indirect provided in the command can be one or two operands based on the atomic operation. Ex: Atomic-Compare operation needs two operands while Atomic-Add operation needs single operand and Atomic-increment requires no operand. Refer "Atomics" sub-section of "L3 Cache and URB" section of the BSpec for detailed atomic operations supported. Atomic operations can be enabled to return value by setting "Return Data Control" field in the command, return data is stored to CS_GPR registers.
CS_GPR4/5 registers are updated with memory Return Data based on the "Data Size". Each GPR register is qword in size and occupies two MMIO registers.
Note: Any references to CS_GPR registers in the command should be understood as the CS_GPR registers belonging to the corresponding engines *CS_GPR registers.

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

Indirect Source Operands:
Operand1 is sourced from [CS_GPR1, CS_GPR0]
Operand2 is sourced from [CS_GPR3, CS_GPR2]
Read return Data is stored in [CS_GPR_5, CS_GPR4]

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

Programming Notes

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

MI_ATOMIC

must be programmed accordingly.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 2Fh MI_ATOMIC Format: OpCode
	22	**Memory Type** This bit will be ignored and treated as if clear when executing from a non-privileged batch buffer. It is allowed for this bit to be clear when executing this command from a privileged (secure) batch buffer. This bit must be 1 if the **Per Process GTT Enable** bit is clear. Value 0h: Per Process Graphics Address Value 1h: Global Graphics Address — This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.
	21	**Reserved** Source: BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS Format: MBZ
	21	**Post-Sync Operation** Source: RenderCS, PositionCS Value 0h: No Post Sync Operation — Command is executed as usual. Value 1h: Post Sync Operation — MI_ATOMIC command is executed as a pipelined PIPE_CONTROL flush command with Atomics operation as post sync operation. Flush completion only guarantees the workload prior to this command is pushed till Windower unit and completion of any outstanding flushes issued prior to this command. When this bit set following restriction apply to atomic operation: • Non-Compare atomic operations are supported on data granularity of 4B and 8B. DW3 is the lower dword of the operand and DW4 is the upper dword of the operand for the atomic

		operation. • Compare atomic operations are supported on data granularity of 4B. DW3 is Operand-0 and DW4 is Operand-1 for the atomic operation. • Atomic operations to GGTT/PPGTT memory surface are supported. • Only Inline data mode for atomic operand is supported, no support for indirect data mode. • No support for Return Data Control functionality. • No support for atomic operations on data granularity of 16B. • No support for compare atomic operations on data granularity of 8B.

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

20:19 **Data Size**

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

Value	Name	Description
0h	DWORD	Operand size used by Atomic Operation is DWORD.
1h	QWORD	Operand Size used by Atomic Operation is QWORD.
2h	OCTWORD	Operand Size used by Atomic Operation is OCTWORD.
3h	RESERVED	

18 **Inline Data**

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

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

17 **CS STALL**

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

next command.

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

	16	**Return Data Control** Source: RenderCS, BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS **Description** When Return Data Control is set the read return feature will be enabled during the atomic operation. Data is stored in CS_GPR5/4 registers unconditionally on completion of the atomic operation. On data return CS_GPR5/4 Registers are updated based on the "Data Size" field. When "Data Size" is DWORD lower dword of CS_GPR4 (Qword) is updated with the dword data returned from memory. When "Data Size" is QWORD only CS_GPR4 (Qword) is updated with the qword data returned from memory. When the data size is OCTWORD CS_GPR4/5 are updated with the OCTWORD data returned from memory. CS_GPR4 is loaded with lower qword returned from memory and CS_GPR5 is loaded with upper qword returned from memory.
	15:8	**ATOMIC OPCODE** This field selects the kind of atomic operation to be performed. Refer "Atomics" sub-section of "L3 Cache and URB" section for atomic opcode encoding and operation.
	7:0	**DWord Length** Format: =n Total Length - 2. Excludes DWord (0,1).

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

1	31:2	**Memory Address** Format: GraphicsAddress[31:2] This field contains the graphics memory address of the data on which atomic operation has to be performed. Atomic operation can be performed on data granularity of 4B, 8B or 16B and hence the Address has to be correspondingly aligned to 4B,8B or 16B respectively.
	1	**Reserved**
	0	**Reserved** Format: MBZ
2	31:16	**Reserved**

MI_ATOMIC

		Format: MBZ
	15:0	**Memory Address High** This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space.
3	31:0	**Operand1 Data Dword 0** Format: U32 Dword0 of Operand1 when Inline Data mode is set.
4	31:0	**Operand2 Data Dword 0** Format: U32 Dword0 of Operand2 when Inline Data mode is set.
5	31:0	**Operand1 Data Dword 1** Format: U32 Dword1 of Operand1 when Inline Data mode is set.
6	31:0	**Operand2 Data Dword 1** Format: U32 Dword1 of Operand2 when Inline Data mode is set.
7	31:0	**Operand1 Data Dword 2** Format: U32 Dword2 of Operand1 when Inline Data mode is set.
8	31:0	**Operand2 Data Dword 2** Format: U32 Dword2 of Operand2 when Inline Data mode is set.
9	31:0	**Operand1 Data Dword 3** Format: U32 Dword3 of Operand1 when Inline Data mode is set.
10	31:0	**Operand2 Data Dword 3** Format: U32 Dword3 of Operand2 when Inline Data mode is set.

MI_BATCH_BUFFER_END

MI_BATCH_BUFFER_END

Source: CommandStreamer

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
	22:16	**Reserved**
	15	**Reserved**
	14:1	**Reserved** Format: MBZ
	0	**End Context** Format: Enable This bit must only be set within a context image. If this command is parsed with this bit set then the engine will consider the context image restore complete. This bit is ignored if parsed during a batch buffer.

MI_BATCH_BUFFER_START

MI_BATCH_BUFFER_START

Source: CommandStreamer

Length Bias: 2

The MI_BATCH_BUFFER_START command is used to initiate the execution of commands stored in a batch buffer. For restrictions on the location of batch buffers, see Batch Buffers in the Device Programming Interface chapter of *MI Functions*. The batch buffer can be specified as privileged or non-privileged, determining the operations considered valid when initiated from within the buffer and any attached (chained) batch buffers. See Batch Buffer Protection in the Device Programming Interface chapter of *MI Functions*.

Programming Notes

- A batch buffer initiated with this command must end either with a MI_BATCH_BUFFER_END command or by chaining to another batch buffer with an MI_BATCH_BUFFER_START command.
- It is essential that the address location beyond the current page be populated inside the GTT. HW performs over-fetch of the command addresses and any over-fetch requires a valid TLB entry. A single extra page beyond the batch buffer is sufficient.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 31h MI_BATCH_BUFFER_START Format: OpCode
	22	**Second Level Batch Buffer** The command streamer contains three storage elements; one for the ring head address, one for the batch head address, and one for the second level batch head address. When performing batch buffer chaining, hardware simply updates the head pointer of the first 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 first level batch buffer address. This allows hardware to mimic a simple 3-level stack. Value / Name / Description 0h / First level batch / Place the batch buffer address in the 1st (traditional) level batch address storage element. 1h / Second level / Place the batch buffer address in the second-level batch

	batch	address storage element.

Programming Notes
Setting this bit when the command is parsed in the ring, will cause the command to be ignored. This bit must only be set in a first level batch buffer.
Chaining of second level batch buffers is supported, i.e MI_BATCH_BUFFER_START command with "Second Level Batch Buffer" bit field set is allowed to be programed as part of the command sequence programed inside a second level batch buffer.

21 **POSH Start**

Source:	RenderCS, PositionCS

Batch buffers dedicated to be executed by POSH pipe are indicated by setting the field “POSH Start” in the MI_BATCH_BUFFER_START command header. Once “POSH Start” is set in a batch buffer all the following chained batch buffers and next level batch buffers will implicitly inherit the “POSH Start” field value. Once “POSH Start” is set in a batch buffer all the following command sequences are to be **executed** by POCS until the corresponding batch buffer sequencing is terminated through MI_BATCH_BUFFER_END/MI_CONDITIONAL_BATCH_BUFFER_END command.
Example:

- Once “POSH Start” is encountered in a first level batch buffer by POCS, it will get reset only when the first level batch buffer execution is terminated through batch buffer end and the command execution sequence goes back to the ring buffer,
- Similarly, once “POSH Start” is encountered in a second level batch buffer by POCS, it will get reset only when the second level batch buffer execution is terminated through batch buffer end and the command execution sequence goes back to the first level buffer,
- Similarly, once when “POSH Start” is encountered in a third level batch buffer by POCS, it will get reset only when the third level batch buffer execution is terminated through batch buffer end and the command execution sequence goes back to the second level batch buffer.

Command sequences executed from the “POSH Start” batch buffer may lead to chained batch buffers or next level batch buffers. Batch buffers executed by POCS may have MI Commands, 3DSATE commands and 3DPRIMTIVE commands for POSH pipe. RCS on parsing MI_BATCH_BUFFER_START command with “POSH Start” enabled NOOPS the command and moves on the following command.

Value	Name	Description
0	**[Default]**	Batch buffer is not "POSH Start" enabled.
1		Batch buffer is "POSH Start" enabled.

Programming Notes
Resource Streamer must not be enabled for "POSH Enable" or "POSH Start" batch buffers.

21:20 **Reserved**

Source:	BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS
Format:	MBZ

20 **POSH Enable**

Source:	RenderCS, PositionCS

"POSH Enable" field in the MI_BATCH_BUFFER_START command is a hint to POCS to traverse (parse, don't execute) the batch buffer to look for "POSH Start" batch buffers. "POSH Enable" field is only inherited to the chained batch buffer and doesn't get inherit to the next level batch buffers unlike "POSH Start" field. "POSH Enable" field must be explicitly set in the MI_BATCH_BUFFER_START command which calls the next level batch buffers in order for the POCS to parse them to look for "POSH Start" batch buffers. "POSH Start" field takes precedence over the "POSH Enable" field in POCS.

Example:

- Once "POSH Enable" is encountered in a first level batch buffer, POCS will traverse the whole of the first level batch buffers (including chained first level) to check for "POSH Start" field in MI_BATCH_BUFFER_START command. POCS by default will not traverse the second level batch buffers. SW must explicitly set the "POSH Enable" field for the second level batch buffer called from first level batch buffer if the second level batch buffer have to be traversed by POCS.
- Similarly, Once "POSH Enable" is encountered in a second level batch buffer, POCS will traverse the whole of the second level batch buffers (including chained second level) to check for "POSH Start" field in MI_BATCH_BUFFER_START command. POCS by default will not traverse the third level batch buffers. SW must explicitly set the "POSH Enable" field for the third level batch buffer called from second level batch buffer if the third level batch buffer have to be traversed by POCS.
- Similarly, Once "POSH Enable" is encountered in a third level batch buffer, POCS will traverse the whole of the third level batch buffers (including chained second level) to check for "POSH Start" field in MI_BATCH_BUFFER_START command.

RCS ignores "POSH Enable" field and has no implications due to the "POSH Enable" field set in the MI_BATCH_BUFFER_START command.

Value	Name	Description

0	**[Default]**	Batch buffer is not "POSH Enable".
1		Batch buffer is "POSH Enable"/

Programming Notes
POCS executes only the MI_BATCH_BUFER_START commands programmed in the ring buffer with "POSH Enable" set and NOOPS (predicates) all the other commands in the ring buffer. POCS only parses/traverses the batch buffer with "POSH Enable" to check for any batch buffer programmed with "POSH Start" set. SW must set "POSH Enable" field in the MI_BATCH_BUFFER_START command programmed in ring buffer if the commands in the corresponding batch buffer or the chained batch buffers (includes Second Level and third level) has at least one batch buffer start command with "POSH Start" set (also implies 3DPRIMITIVE command for which POSH is enabled).
Resource Streamer must not be enabled for "POSH Enable" or "POSH Start" batch buffers.

19 **Enable Command Cache**

Source:	RenderCS

Command Buffer DMA engine on processing the MI_BATCH_BUFFER_START command with "Enable Command Cache" set will make the corresponding command buffer read requests cacheable in L3, it also applies the command caching to all subsequent chained and next level batch buffers. Command buffer DMA engine uses the "Command Buffer Cache Size" programmed in the CMD_BUF_CCTL register to limit the read requests of a cacheable batch buffer to be cached in L3. DMA engine does this by tracking the amount of read requests made cacheable and stops caching when the read requested data size equals to the size of the command cache allocated. This avoids thrashing of cache for Batch Buffers larger in size compared to the command buffer cache allocated in L3.
DMA engine resets the command caching tracker on events listed below.

- On an End Of Tile in PTRBR/POSH mode of operation.
- On a context save of a context.
- On command cache invalidation through PIPE_CONTROL.

Command buffer caching must not be enabled for batch buffers in GGTT address space.
For best performance in terms of utilizing the command buffer section of L3 cache for PTBR mode, the recommendation is to ensure command buffers not to cross 128KB boundary. Avoiding a large overlap around 128KB boundary will minimize the cross distribution of 1KB chunks randomly between the L3 banks creating hotspots otherwise.

Value	Name	Description

1		Command Cache enabled
0	**[Default]**	Command Cache disable

18 **Reserved**

Format:	MBZ

17 **Reserved**

16 **Reserved**

Source:	BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS
Format:	MBZ

16 **Reserved**

Source:	RenderCS
Format:	MBZ

15 **Reserved**

Source:	BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS
Format:	MBZ

15 **Predication Enable**

Source:	RenderCS, PositionCS

This bit is used to enable predication of this command. If this bit is set and Bit 0 of the MI_PREDICATE_RESULT_1 register is clear, this command is ignored. Otherwise the command is performed normally.

14:11 **Reserved**

Format:	MBZ

10 **Reserved**

Format:	MBZ

9 **Reserved**

Source:	RenderCS, BlitterCS
Format:	MBZ

9 **Reserved**

8 **Address Space Indicator**
Batch buffers accessed via PPGTT are considered as non-privileged. Certain

operations (e.g., MI_STORE_DATA_IMM commands to GGTT memory) are prohibited within non-privileged buffers. More details mentioned in User Mode Privileged command section. When MI_BATCH_BUFFER_START command is executed from within a batch buffer (i.e., is a "chained" or "second level" batch buffer command), the current active batch buffer's "Address Space Indicator" and this field determine the "Address Space Indicator" of the next buffer in the chain.

- Chained or Second level batch buffer can be in GGTT or PPGTT if the parent batch buffer is in GGTT.
- Chained or Second level batch buffer can only be in PPGTT if the parent batch buffer is in PPGTT. This is enforced by Hardware.

Value	Name	Description
0h	GGTT	This batch buffer is located in GGTT memory and is privileged.
1h	PPGTT	This batch buffer is located in PPGTT memory and is Non-Privileged.

Programming Notes
This field must be '0' unless the Per-Process GTT Enable is '1'

7:0 **DWord Length**

Default Value:	1h Excludes DWord (0,1)
Format:	=n

Total - Bias. Excludes DWord (0,1).

1..2 GraphicsAddress is a 64-bit value [63:0], but only a portion of it is used by hardware. The upper reserved bits are ignored and MBZ.

63:2 **Batch Buffer Start Address**

Format:	GraphicsAddress63-2

1:0 **Reserved**

Format:	MBZ

MI_CLFLUSH

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 and is not privileged.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 27h Store DW MI_CLFLUSH Format: OpCode
	22	**Use Global GTT** 0h – Per Process Graphics Address 1h – Global Graphics Address – This command will use the global GTT to translate the Address.
	21:10	**Reserved** Format: MBZ
	9:0	**DWord Length** Format: n Total Length - 2 1h – **[Default]** – Excludes DWord (0,1)
1..2	63:12	**Page Base Address** Format: GraphicsAddress63-12 4KB aligned Page Address for which software requires a hardware flush to DRAM. GraphicsAddress is 64-bit value [63:0], but only a portion of it is used by the hardware. The uppermost bits are ignored and MBZ. Some GraphicsAddress fields only specify the upper address bits. For example GraphicsAddress[47:12] is a 4KB page address.
	11:6	**Starting Cacheline Offset** Format: U6

<table>
<tr><th colspan="3">MI_CLFLUSH</th></tr>
<tr><td></td><td></td><td>Zero based starting cacheline offset in to the Page Base Address.</td></tr>
<tr><td></td><td>5:0</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>3..n</td><td>31:0</td><td>DW Representing a Half Cache Line<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

MI_CONDITIONAL_BATCH_BUFFER_END

Source: CommandStreamer

Length Bias: 2

Description

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

Any updates to the memory location exercised by this command must be ensured to be coherent in memory prior to programming of this command. If the memory location is being updated by a prior executed MI command (ex: MI_STORE_REGISTER_MEM ..etc) on the same engine, SW must follow one of the below programming note prior to programming of this command to ensure data is coherent in memory.
Option1: Programming of "4" 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 (4 times) (Dummy data, Scratch Address) MI_CONDITIONAL_BATCH_BUFFER_END(0x2CF0_0000)
Option2: Programming of a PIPE_CONTROL with Post-Sync Operation selected to “Write Immediate Data” to scratch space address with “Command Streamer Stall Enable” set prior to programming of this command. Example: MI_STORE_REGISTE_MEM (0x2288, 0x2CF0_0000) .... PIPE_CONTROL (Stall, Write Immediate Data), MI_CONDITIONAL_BATCH_BUFFER_END(0x2CF0_0000).
Option3: MI_ATOMIC (write to scratch space) with "CS STALL" set prior to programming of this command. Example: MI_STORE_REGISTE_MEM (0x2288, 0x2CF0_0000) .... MI_ATOMIC (MOV, Dummy data, Scratch Address), MI_CONDITIONAL_BATCH_BUFFER_END(0x2CF0_0000).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 36h MI_CONDITIONAL_BATCH_BUFFER_END Format: OpCode
	22	**Use Global GTT** Default Value: 0h Format: Boolean If set, this command will use the global GTT to translate the **Compare Address** and this command must be executing from a privileged (secure) batch buffer. If clear, the PPGTT will be used to translate the **Compare Address**.

MI_CONDITIONAL_BATCH_BUFFER_END

	21	**Compare Semaphore** Format: Boolean This bit provides a means to enable or disable compare operation. **Value** / **Name** / **Description** 0h / / This command will be treated as NOOP and usual batch buffer execution flow continues. 1h / **[Default]** / The data from the "Compare Data Dword" (inline) is compared to the data in memory pointed by the "Compare Address". If the value at Compare Address is greater than the Compare Data Dword, execution of the batch buffer will continue, else the execution of all levels of the batch buffer are terminated and execution of the command sequence returns to the ring buffer.
	20	**Reserved**
	19	**Compare Mask Mode** **Value** / **Name** / **Description** 0h / Compare Mask Mode Disabled / Compare address points to Dword in memory consisting of Data Dword(DW0). HW will compare Data Dword(DW0) against Semaphore Data Dword. 1h / Compare Mask Mode Enabled / Compare address points to Qword in memory consisting of compare Mask (DW0) and Data Dword(DW1). HW will do AND operation on Mask(DW0) with Data Dword(DW1) and then compare the result against Semaphore Data Dword. **Programming Notes** When "Compare Mask Mode" is enabled, "Compare Address" must be qword aligned.
	18	**Reserved**
	17:16	**Reserved**
	15	**Reserved**
	14:12	**Reserved**
	11:8	**Reserved** Format: MBZ

<table>
<tr><th colspan="3">MI_CONDITIONAL_BATCH_BUFFER_END</th></tr>
<tr><td></td><td>7:0</td><td>DWord Length<table><tr><td>Default Value:</td><td>2h Excludes DWord (0,1)</td></tr><tr><td></td><td></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>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.</td></tr>
<tr><td rowspan="2">2..3
Qword address to fetch Data Qword from memory. This field specifies the 4GB aligned base address of Gfx 4GB virtual address space within the host's 64-bit virtual address space. GraphicsAddress is a 64-bit value [63:0], but only a portion of it is used by hardware. Upper reserved bits are ignored and MBZ.</td><td>63:3</td><td>Compare Address<table><tr><td>Format:</td><td>GraphicsAddress63-3</td></tr></table></td></tr>
<tr><td>2:0</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
</table>

MI_COPY_MEM_MEM

MI_COPY_MEM_MEM

Source: VideoCS

Length Bias: 2

The MI_COPY_MEM_MEM command reads a DWord from memory and stores the value of that DWord to back to memory. The source and destination addresses are specified in the command. The command temporarily halts command execution.

Programming Notes

This command should not be used within a "non-privileged" batch buffer to access global virtual space; doing so will be treated as privilege access violation. Refer to the "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to learn more about HW behavior on encountering a privilege access violation.
This command can be used within ring buffers and/or privilege batch buffers to access global virtual space.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 2Eh MI_MEM_TO_MEM Format: OpCode
	22	**Use Global GTT Source** It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear. 0h – Per Process Graphics Address 1h – Global Graphics Address: This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.
	21	**Use Global GTT Destination** It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear. 0h – Per Process Graphics Address

		1h	Global Graphics Address	This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.
	20:8	**Reserved**		
		Format:		MBZ
	7:0	**DWord Length**		
		Default Value:		3 Excludes DWord (0,1)
		Format:		=n Total Length - 2
1..2	63:2	**Destination Memory Address**		
		Format:		GraphicsAddress63-2
		Surface Type: MMIO Register		
	1:0	**Reserved**		
		Format:		MBZ
3..4	63:2	**Source Memory Address**		
		Format:		GraphicsAddress63-2
		Surface Type: MMIO Register		
	1:0	**Reserved**		
		Format:		MBZ

MI_COPY_MEM_MEM

MI_COPY_MEM_MEM

Source: VideoEnhancementCS

Length Bias: 2

The MI_COPY_MEM_MEM command reads a DWord from memory and stores the value of that DWord to back to memory. The source and destination addresses are specified in the command. The command temporarily halts command execution.

Programming Notes

This command should not be used within a "non-privileged" batch buffer to access global virtual space; doing so will be treated as privilege access violation. Refer to the "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to learn more about HW behavior on encountering a privilege access violation.
This command can be used within ring buffers and/or privilege batch buffers to access global virtual space.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 2Eh MI_MEM_TO_MEM Format: OpCode
	22	**Use Global GTT Source** It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear. 0h – Per Process Graphics Address 1h – Global Graphics Address: This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.
	21	**Use Global GTT Destination** It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear. 0h – Per Process Graphics Address

		1h	Global Graphics Address	This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.
	20:8	**Reserved**		
		Format:		MBZ
	7:0	**DWord Length**		
		Default Value:		3 Excludes DWord (0,1)
		Format:		=n Total Length - 2
1..2	63:2	**Destination Memory Address**		
		Format:	GraphicsAddress63-2	
		Surface Type: MMIO Register		
	1:0	**Reserved**		
		Format:		MBZ
3..4	63:2	**Source Memory Address**		
		Format:	GraphicsAddress63-2	
		Surface Type: MMIO Register		
	1:0	**Reserved**		
		Format:		MBZ

MI_COPY_MEM_MEM

MI_COPY_MEM_MEM

Source: BlitterCS

Length Bias: 2

The MI_COPY_MEM_MEM command reads a DWord from memory and stores the value of that DWord to back to memory. The source and destination addresses are specified in the command. The command temporarily halts command execution.

Programming Notes

This command should not be used within a "non_privilege" batch buffer to access global virtual space, doing so will be treated as privilege access violation. Refer "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation. This command can be used within ring buffers and/or privilege batch buffers to access global virtual space.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 2Eh MI_COPY_MEM_MEM Format: OpCode
	22	**Use Global GTT Source** It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear. Value / Name / Description: 0h — Per Process Graphics Address 1h — Global Graphics Address — This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.
	21	**Use Global GTT Destination** It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear. Value / Name / Description:

		0h: Per Process Graphics Address 1h: Global Graphics Address — This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.
	20:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 3 Excludes DWord (0,1) Format: =n Total Length - 2
1..2 Surface Type: MMIO Register GraphicsAddress is a 64-bit value [63:0], but only a portion of it is used by hardware. The upper reserved bits are ignored and MBZ.	63:0	**Destination Memory Address** Format: GraphicsAddress63-0
3..4 Surface Type: MMIO Register GraphicsAddress is a 64-bit value [63:0], but only a portion of it is used by hardware. The upper reserved bits are ignored and MBZ.	63:0	**Source Memory Address** Format: GraphicsAddress63-0

MI_COPY_MEM_MEM

MI_COPY_MEM_MEM

Source: RenderCS

Length Bias: 2

The MI_COPY_MEM_MEM command reads a DWord from memory and stores the value of that DWord to back to memory. The source and destination addresses are specified in the command. The command temporarily halts command execution.

Programming Notes

This command should not be used within a "non_privilege"batch buffer to access global virtual space, doing so will be treated as privilege access violation. Refer "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation. This command can be used within ring buffers and/or privilege batch buffers to access global virtual space.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 2Eh MI_COPY_MEM_MEM Format: OpCode
	22	**Use Global GTT Source** It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear. Value 0h: Per Process Graphics Address Value 1h: Global Graphics Address — It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear.
	21	**Use Global GTT Destination** This bit will be ignored and treated as if clear when executing from a non-privileged batch buffer. It is allowed for this bit to be clear when executing this command from a privileged (secure) batch buffer. This bit *must* be '1' if the **Per Process GTT Enable** bit is clear. This bit will determine write to memory uses Global or Per Process GTT. Value / Name / Description

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

	20:8	**Reserved**
		Format: MBZ
	7:0	**DWord Length**
		Default Value: 3 Excludes DWord (0,1)
		Format: =n Total Length - 2
1..2	63:2	**Destination Memory Address**
		Format: GraphicsAddress[63:2]
		Surface Type: MMIO Register This field specifies the address of the memory location where the value fetched specified in the DWord address above will be written. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[63:2] for a DWord register GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].
	1:0	**Reserved**
		Format: MBZ
3..4	63:2	**Source Memory Address**
		Format: GraphicsAddress[63:2]
		Surface Type: MMIO Register This field specifies the address of the memory location where the value is located that will be written to the address below. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[63:2] for a DWord register GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].
	1:0	**Reserved**
		Format: MBZ

MI_DISPLAY_FLIP

MI_DISPLAY_FLIP

Source: RenderCS, 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 (possibly on a future frame), or asynchronously (as soon as possible) to minimize rendering stalls at the cost of tearing artifacts.

Programming Notes

1. 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 command does not guarantee that outstanding flip operations have completed). The MI_WAIT_FOR_EVENT command can be used to provide this synchronization - 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.
2. After a display flip operation is requested, software is responsible for initiating any required synchronization with subsequent buffer clear or rendering operations. For multi-buffering (e.g., double buffering) operations, this will typically require updating SURFACE_STATE or the binding table to change the rendering (back) buffer. In addition, prior to any subsequent clear or rendering operations, software must typically ensure that the new rendering 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.
3. 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.
4. 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.
5. DWord 3 (Left Eye Display Buffer Base Address) must not be set with synchronous flips or asynchronous

MI_DISPLAY_FLIP

flips. It is only allowed to be sent with stereo 3D flips.

Asynchronous flip completion time depends greatly on how much data has been prefetched for power savings, and can take up to 1 full frame to complete. For faster flip completion, disable FBC and render compression and allocate a small amount of data buffer for the plane.

"Command Streamer Plane Number" mapping to "Display Plane Name" are listed in display B-spec -"Plane capability and Interoperability".

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 14h MI_DISPLAY_FLIP Format: OpCode
	22	**Async Flip Indicator** Format: Enable This bit should always be set if DW2 [1:0] == '01' (async flip). This field is required due to HW limitations. This bit is used by the render pipe while DW2 is used by the display hardware.
	21:19	**Reserved** Format: MBZ
	18:17	**Reserved**
	16:14	**Reserved** Format: MBZ
	13:8	**Display Plane Select** Value / Name: 0h Display Plane 1 1h Display Plane 2 2h Display Plane 3 3h Reserved 4h Display Plane 4 5h Display Plane 5 6h Display Plane 6 7h Display Plane 7 8h Display Plane 8

		9h	Display Plane 9
		Ah	Display Plane 10
		Bh	Display Plane 11
		Ch	Display Plane 12
		Dh	Display Plane 13
		Eh	Display Plane 14
		Fh	Display Plane 15
		10h	Display Plane 16
		11h	Display Plane 17
		12h	Display Plane 18
		13h	Display Plane 19
		14h	Display Plane 20
		15h	Display Plane 21
		16h	Display Plane 22
		17h	Display Plane 23
		18h	Display Plane 24
		19h	Display Plane 25
		1Ah	Display Plane 26
		1Bh	Display Plane 27
		1Ch	Display Plane 28
		1Dh	Display Plane 29
		1Eh	Display Plane 30
		1Fh	Display Plane 31
		20h	Display Plane 32
		[21h, 3Fh]	Reserved
	7:0	**DWord Length**	
		Format:	=n
		Total Length - 2. Excludes DWord (0,1). For Synchronous Flips and Asynchronous Flips, this field must be programmed to 1h for a total length of 3. For Stereo 3D Flips, this field must be programmed to 2h for a total length of 4.	
		Value / **Name** / **Exists If**	
		1h / **[Default]** / ([Flip Type]!='Stereo 3D Flip')	
		2h / **[Default]** / ([Flip Type]=='Stereo 3D Flip')	
1	31	**Stereoscopic 3D Mode**	
		Default Value:	0h

		Format: Enable This bit must be set if the Extra Display Buffer Address is part of this command. This bit is used to notify the display there is an extra DW before processing the Display Flip.
	30:16	**Reserved** Format: MBZ
	15:6	**Display Buffer Pitch** Default Value: 0h Format: U10 *For Synchronous Flips and Stereo 3D Flips only,* this field specifies the pitch of the new display buffer. For Asynchronous Flips, this parameter is programmed so that all the flips in a flip chain should maintain the same pitch as programmed with the last synchronous flip or stereo 3D flip or direct through MMIO. See the Display Plane Stride register for details.
	5:3	**Reserved** Format: MBZ
	2:0	**Tile Parameter** Format: Enable 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 through MMIO.

Value	Name	Description
0h	Linear **[Default]**	For Synchronous Flips Only
1h	Tiled X	
2h-3h	Reserved	
4h	Tiled Y Legacy (Y B)	
5h	Tiled Y F	

2	31:12	**Display Buffer Base Address** Format: GraphicsAddress[31:12] This field specifies Bits 31:12 of the Graphics Address of the new display buffer. In stereo 3D mode this is the right eye base address. In non-stereo 3D mode this is the only base address. (Refer to the Display Address Start Address Register description in the Display Registers chapter).

Programming Notes

- The Display buffer must reside completely in Main Memory.
- This address is always translated via the global (rather than per-process) GTT.

	11	**Reserved**

		Format: MBZ
	10:7	**Reserved** Format: MBZ
	6:4	**Reserved**
	3	**Reserved** Format: MBZ
	2	**Reserved**
	1:0	**Flip Type** This field specifies whether the flip operation should be performed asynchronously to vertical retrace.

Value	Name	Description
00b	Sync Flip **[Default]**	The flip will occur during the vertical blanking interval - thus avoiding any tearing artifacts.
01b	Async Flip	The flip will occur "as soon as possible" - and may exhibit tearing artifacts
10b	Stereo 3D Flip	The flip will occur during the vertical blanking interval (left or right eye blank selectable through display MMIO register) - thus avoiding any tearing artifacts.
11b	Reserved	

Programming Notes

- The Display Buffer Pitch and Tile parameter cannot be changed for asynchronous flips (i.e., the new buffer must have the same pitch/tile format as the previous buffer). Only display surface base address can be changed.
- For Asynch Flips the Buffers used must be 32KB aligned.
- Asynch flips are supported on primary or universal planes only.

- For Stereo 3D flips, both left and right eye buffers must have the same pitch and tile format.

3	31:12	**Left Eye Display Buffer Base Address** Format: GraphicsAddress[31:12] This field specifies Bits 31:12 of the Graphics Address of the new display buffer for the stereo 3D left eye. In non-stereo 3D mode this address is not used. (Refer to the Display Address Start Address Register description in the Display Registers chapter).

Programming Notes

- The Display buffer must reside completely in Main Memory.
- This address is always translated via the global (rather than per-process) GTT.

MI_DISPLAY_FLIP			
	11:0	**Reserved**	
		Format:	MBZ

MI_FLUSH_DW

MI_FLUSH_DW

Source: VideoEnhancementCS

Length Bias: 2

The MI_FLUSH_DW command is used to perform an internal "flush" operation. The parser pauses on an internal flush until all drawing engines have completed any pending operations. In addition, this command can also be used to:

- Flush any dirty data to memory.
- Invalidate the TLB cache inside the hardware

Usage note: After this command is completed with a Store DWord enabled, CPU access to graphics memory will be coherent (assuming the Render Cache flush is not inhibited).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 26h MI_FLUSH_DW Format: OpCode
	22	**Reserved**
	21	**Store Data Index** Format: U1 **Description** This field is valid only if the post-sync operation is not 0. If this bit is set, the store data address is index into the global hardware status page when destination address type in the command is set to 1 (GGTT). The store data address is index into the per-process hardware status page when destination address type in the command is set to 0 (PPGTT).
	20:19	**Reserved** Format: MBZ
	18	**TLB Invalidate** Format: U1

Description
If ENABLED, all TLBs belonging to Video Enhancement Engine will be invalidated once the flush operation is complete. This bit is only valid when the Post-Sync Operation field is a value of 1h or 3h.
If GFX_MODE (0x229c) bit 13, this command will cause a config write to MMIO register space with the address 0x4f100.

17 **Reserved**

Format:	MBZ

16 **Reserved**

Format:	MBZ

15:14 **Post-Sync Operation**

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	Reserved	Reserved
3h	Write TIMESTAMP register	Write the TIMESTAMP register to the Destination Address. The upper 28 bits of the TIMESTAMP register are tied to '0'.

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

13:10 **Reserved**

Format:	MBZ

9 **Flush LLC**

Format:	Enable

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

8 **Notify Enable**

		Format: U1
		If ENABLED, a Sync Completion Interrupt will be generated (if enabled by the MI Interrupt Control registers) once the sync operation is complete. See Interrupt Control Registers in Memory Interface Registers for details.
	7	**Reserved** Format: MBZ
	6	**Reserved** Format: MBZ
	5:0	**DWord Length** Format: =n Total Length - 2 3h: Excludes DWord (0,1) = 2 for DWord, 3 for QWord **[Default]**
1	31:3	**Address** Format: GraphicsAddress[31:3]U28 This field specifies Bits 31:3 of the Address where the DWord or QWord will be stored. Note that the address can only be QWord aligned, irrespective of data size.
	2	**Destination Address Type** Defines address space of Destination Address 0h: PPGTT — Use PPGTT address space for DW write 1h: GGTT — Use GGTT address space for DW write **Programming Notes** Ignored if "No write" is the selected in Operation.
	1:0	**Reserved** Format: MBZ
2	31:16	**Reserved**

		Format: MBZ
	15:0	**Address High** Format: GraphicsAddress[47:32]U64 This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space
3..4	63:0	**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_FLUSH_DW

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 Format: OpCode
	28:23	**MI Command Opcode** Default Value: 26h MI_FLUSH_DW Format: OpCode
	22	**Reserved** Format: U1
	21	**Store Data Index** Format: U1 **Description** This field is valid only if the post-sync operation is not 0. If this bit is set, the store data address is index into the global hardware status page when destination address type in the command is set to 1 (GGTT). The store data address is index into the per-process hardware status page when destination address type in the command is set to 0 (PPGTT).
	20:19	**Reserved** 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	**Reserved** Format: MBZ
	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.
1h	Write Immediate Data QWord	Write the QWord containing Immediate Data Low, High DWs to the Destination Address
2h	Reserved	Reserved
3h	Write TIMESTAMP Register	Write the TIMESTAMP register to the Destination Address. The upper 28 bits of the TIMESTAMP register are tied to '0'.

Programming Notes
If executed in a PPGTT (non-secure) batch buffer, the post-sync op is always inhibited. This command does not write anything out to memory. [For all other devices]: If executed in a non-secure batch buffer, the address given is in a PPGTT address space. If in a secure ring or batch, the address given is in GGTT space.

	13:10	**Reserved**

		Format: MBZ
	9	**Flush LLC** Format: Enable If enabled, at the end of the current MI_FLUSH_DW the last level cache is cleared of all the cachelines which have been determined as being part of the Frame Buffer.
	8	**Notify Enable** Format: U1 If ENABLED, a Sync Completion Interrupt will be generated (if enabled by the MI Interrupt Control registers) once the sync operation is complete. See Interrupt Control Registers in Memory Interface Registers for details.
	7:6	**Reserved** Format: MBZ
	5:0	**DWord Length** Format: =n Total Length - 2 Value 3h: Excludes DWord (0,1) = 2 for DWord, 3 for QWord **[Default]**
1..2 This field specifies the destination address where the DWord or QWord will be stored. Note that the address can only be QWord aligned, irrespective of data size. GraphicsAddress is 64-bit value [63:0], but only a portion of it is used by hardware. The upper reserved bits are ignored and MBZ. Some GraphicsAddress fields only specify the upper address bits. For example GraphicsAddress[47:12] would be a 4KB page address.	63:48	**Reserved** Format: MBZ
	47:3	**Destination Address** Format: GraphicsAddress[47:3]
	2	**Destination Address Type** Defines the address space of the Destination Address. Value 0h: PPGTT — Use PPGTT address space for DW write

		1h | GGTT | Use GGTT address space for DW write **Programming Notes** Ignored if "No write" is the selected in Operation.
	1:0	**Reserved** Format: MBZ
3..4	63:0	**Immediate Data** This field specifies the DWord value to be written to the targeted location. Only valid when 15:14 in header is set to 1h. **Programming Notes** To avoid hitting a known hardware bug, drivers cannot send a QW write when bit 5 of the address is '1'

MI_FLUSH_DW

MI_FLUSH_DW

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 Format: OpCode
	28:23	**MI Command Opcode** Default Value: 26h MI_FLUSH_DW Format: OpCode
	22	**Reserved**
	21	**Store Data Index** Format: U1 **Description** This field is valid only if the post-sync operation is not 0. If this bit is set, the store data address is index into the global hardware status page when destination address type in the command is set to 1 (GGTT). The store data address is index into the per-process hardware status page when destination address type in the command is set to 0 (PPGTT).
	20:19	**Reserved** Format: MBZ
	18	**TLB Invalidate** Format: U1 If ENABLED, all TLBs belonging to Video Engine will be invalidated once the flush operation is complete. This bit is only valid when the Post-Sync Operation field is a value of 1h or 3h.
	17	**Reserved**

Format:	MBZ

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.
1h	Write Immediate Data	HW implicitly detects the Data size to be Qword or Dword to be written to memory based on the command dword length programmed . When Dword Length indicates Qword, Writes the QWord containing Immediate Data Low, High DWs to the Destination Address . When Dword Length indicates Dword, Writes the DWord containing Immediate Data Low to the Destination Address
2h	Reserved	Reserved
3h		Write the TIMESTAMP register to the Destination Address. The upper 28 bits of the TIMESTAMP register are tied to '0'.

Programming Notes
If executed in a PPGTT (non-secure) batch buffer, the post-sync op is always inhibited. This command does not write anything to memory.

13:10 Reserved

Format:	MBZ

9 Flush LLC

Format:	Enable

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

8 Notify Enable

Format:	U1

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

MI_FLUSH_DW

	7	**Video Pipeline Cache invalidate** Format: U1 Enable the invalidation of the video cache at the end of this flush
	6	**Reserved** Format: MBZ
	5:0	**DWord Length** Format: =n Total Length - 2. Excludes DWord (0,1) 2h DWord 3h QWord **[Default]**
1..2	63:48	**Reserved** Format: MBZ
	47:3	**Address** Format: GraphicsAddress[47:3]U64 This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space. GraphicsAddress is a 64-bit value [63:0], but only a portion of it is used by the hardware. Upper bits [63:48] are ignored and MBZ. Some GraphicsAddress fields only specify the upper address bits. For example GraphicsAddress[47:12] is a 4KB page address.
	2	**Destination Address Type** Defines address space of Destination Address 0h PPGTT Use PPGTT address space for DW write 1h GGTT Use GGTT address space for DW write **Programming Notes** Ignored if "No write" is the selected in Operation.
	1:0	**Reserved** Format: MBZ

MI_FLUSH_DW		
3..4	63:0	**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_FORCE_WAKEUP

MI_FORCE_WAKEUP

Source: CommandStreamer

Length Bias: 2

This command is used to communicate Force Wakeup request to PM unit. No functionality is performed by this command when none of the mask bits are set and is equivalent to NOOP. Example for usage model: VCS Ring Buffer: MI_FORCE_WAKEUP (Force Render Awake set to '1') MI_SEMPAHORE_SIGNAL (Signal context id 0xABC to Render Command Streamer) MI_FORCE_WAKEUP (Force Render Awake set to '0') MI_BATCH_BUFFER_START STATE Commands .. .. MI_FORCE_WAKEUP (Force Render Awake set to '1') MI_LOAD_REGISTER_IMMEDIATE (Load register 0x23XX in render command streamer with data 0xFFF) MI_FORCE_WAKEUP (Force Render Awake set to '0') ..... MI_BATCH_BUFFER_END

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 1Dh MI_FORCE_WAKEUP Format: OpCode
	22:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Format: =n Total Length - 2. Excludes DWord (0,1).
1	31:16	**Mask Bits** Format: Mask[15:0] **Programming Notes** Must be set to modify corresponding Bits 15:0. (Mask bits must not be set for reserved bits).
	15:10	**Reserved** Format: MBZ
	9:8	**Reserved** Format: MBZ
	7:5	**Reserved**

	Format:	MBZ

4	**Force Media-Slice3 Awake**

Format:	U1

When set, Command Streamer sends message to PM to force awake the media engines in media slice 3, i.e., VCS0, VCS1, and VECS0 (next instructions require media engine awake). Command streamer waits for acknowledge from PM before parsing the next command. When reset, command streamer sends message to PM to disable force awake of media engine (next instructions do not require the media engine to be awake). Command streamer waits for acknowledge from PM before parsing the next command.

Programming Notes
Mask bit [20] has to be set for HW to look at this field when MI_FORCE_WAKEUP command is parsed.

3	**Force Media-Slice2 Awake**

Format:	U1

When set, Command Streamer sends message to PM to force awake the media engines in media slice 2, i.e., VCS0, VCS1, and VECS0 (next instructions require media engine awake). Command streamer waits for acknowledge from PM before parsing the next command. When reset, command streamer sends message to PM to disable force awake of media engine (next instructions do not require the media engine to be awake). Command streamer waits for acknowledge from PM before parsing the next command.

Programming Notes
Mask bit [19] has to be set for HW to look at this field when MI_FORCE_WAKEUP command is parsed.

2	**Force Media-Slice1 Awake**

Format:	U1

When set, Command Streamer sends message to PM to force awake the media engines in media slice 1, i.e., VCS0, VCS1, and VECS0 (next instructions require media engine awake). Command streamer waits for acknowledge from PM before parsing the next command. When reset, command streamer sends message to PM to disable force awake of media engine (next instructions do not require the media engine to be awake). Command streamer waits for acknowledge from PM before parsing the next command.

Programming Notes
Mask bit [18] has to be set for HW to look at this field when MI_FORCE_WAKEUP command is parsed.

1	**Force Render Awake**

Source:	BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS
Format:	U1

MI_FORCE_WAKEUP

Description
When set, Command Streamer sends message to PM to force awake render engine (next instructions require render engine awake). Command streamer waits for acknowledge from PM before parsing the next command. When reset, command streamer sends message to PM to disable force awake of render engine (next instructions do not require the render engine to be awake. Command streamer waits for acknowledge from PM before parsing the next command.

Programming Notes
Mask bit [17] has to be set for HW to look at this field when MI_FORCE_WAKEUP command is parsed.
MI_FORCE_WAKEUP command programmed in RenderCS command buffer must not set "Force Render Awake" bit.

0 **Force Media-Slice0 Awake**

Format:	U1

When set, Command Streamer sends message to PM to force awake the media engines in media slice 0, i.e., VCS0, VCS1, and VECS0 (next instructions require media engine awake). Command streamer waits for acknowledge from PM before parsing the next command. When reset, command streamer sends message to PM to disable force awake of media engine (next instructions do not require the media engine to be awake). Command streamer waits for acknowledge from PM before parsing the next command.

Programming Notes
Mask bit [16] has to be set for HW to look at this field when MI_FORCE_WAKEUP command is parsed.

MI_LOAD_REGISTER_IMM

MI_LOAD_REGISTER_IMM

Source: CommandStreamer

Length Bias: 2

The MI_LOAD_REGISTER_IMM command requests a write of up to a DWord constant supplied in the command to the specified Register Offset (i.e., offset into Memory-Mapped Register Range).
Any offset that is to a destination outside of the GT core will allow the parser to continue once the cycle is at the GT boundary and not destination. Any other address will ensure the destination is updated prior to parsing the next command

Programming Notes	Source
Many MMIO bits require the engine to be IDLE prior to updating the value. Command streamer does not implicitly put in a pipeline flush in the cases a MMIO bit requires the engine to be IDLE. In the case there was a 3DPRIMITIVE command or GPGPU_WALKER command without any stalling PIPE_CONTROL, one must be inserted prior to a MI_LOAD_REGISTER_IMMEDIATE that is updating a bit that requires the engine to be IDLE.	RenderCS
When executed from a non-privileged batch buffer, MMIO writes are only allowed to the registers listed in User Mode Non-Privileged Registers for the corresponding engine, any writes targeting the register not listed in the User Mode Non-Privileged Register will convert this command to a NOOP.	
The following addresses should NOT be used for LRIs: 1. 0x8800 - 0x88FF 2. >= 0xC0000 Limited LRI cycles to the Display Engine (0x40000-0xBFFFF) are allowed, but must be spaced to allow only one pending at a time. This can be done by issuing an SRM to the same address immediately after each LRI.	
Programming an MMIO register is equivalent to programming a non-pipeline state to the hardware and hence an explicit stalling flush needs to be programmed prior to programming this command. However for certain MMIO registers based on their functionality doing an explicit stalling flush is exempted. Listed below are the exempted registers. • 3DPRIM_END_OFFSET - Auto Draw End Offset • 3DPRIM_START_VERTEX - Load Indirect Start Vertex • 3DPRIM_VERTEX_COUNT - Load Indirect Vertex Count • 3DPRIM_INSTANCE_COUNT - Load Indirect Instance Count • 3DPRIM_START_INSTANCE - Load Indirect Start Instance • 3DPRIM_BASE_VERTEX - Load Indirect Base Vertex • 3DPRIM_XP0 - Load Indirect Extended Parameter 0 • 3DPRIM_XP1 - Load Indirect Extended Parameter 1 • 3DPRIM_XP2 - Load Indirect Extended Parameter 2	RenderCS

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td>Command Type
<table>
<tr><td>Default Value:</td><td>0h MI_COMMAND</td></tr>
<tr><td>Format:</td><td>OpCode</td></tr>
</table></td></tr>
<tr><td></td><td>28:23</td><td>MI Command Opcode
<table>
<tr><td>Default Value:</td><td>22h MI_LOAD_REGISTER_IMM</td></tr>
<tr><td>Format:</td><td>OpCode</td></tr>
</table></td></tr>
<tr><td></td><td>22:20</td><td>Reserved
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>19</td><td>Add CS MMIO Start Offset
<table>
<tr><td></td><td></td></tr>
</table>
This bit controls the functionality of the "Register Address" field in the command.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>1</td><td></td><td>"Register Address" field in the command is treated as an offset from the executing Command Streamer’s MMIO start offset. Bits [22:2] of the "Register Address" are considered as dword offset to be added to the MMIO start offset of the corresponding command streamer. However, during context restore bits [11:2] of the "Register Address" are considered as dword offset to be added to the MMIO start offset of the corresponding command streamer.
// Command executed from Ring Buffer or Batch Buffer
Example: MI_LOAD_REGISTER_IMMEDIATE, ADD_CS_MMIO_START_OFFSET: true, Data:0xABCD, Register Address: 0x00_2000
The above command when executed on RenderCS will result in a write to MMIO offset 0x00_4000 (0x00_2000 + 0x00_2000) instead to 0x00_2000. Note that RenderCS MMIO start offset is 0x2000. Table below shows the result of this command executed by various command streamers.
<table>
<tr><th>S.No</th><th>Command Streamer</th><th>Command Streamer MMIO Base</th><th>LRI Register Offset</th><th>LRI Written Address</th></tr>
<tr><td>1</td><td>RenderCS</td><td>0x2000</td><td>0x2000</td><td>0x00_4000</td></tr>
<tr><td>2</td><td>VideoCS0</td><td>0x1C_0000</td><td>0x2000</td><td>0x1C_2000</td></tr>
<tr><td>3</td><td>VideoCS1</td><td>0x1C_4000</td><td>0x2000</td><td>0x1C_6000</td></tr>
<tr><td>4</td><td>VideoEnhancement0</td><td>0x1C_8000</td><td>0x2000</td><td>0x1C_A000</td></tr>
<tr><td>5</td><td>VideoCS2</td><td>0x1D_0000</td><td>0x2000</td><td>0x1D_2000</td></tr>
<tr><td>6</td><td>VideoCS3</td><td>0x1D_4000</td><td>0x2000</td><td>0x1D_6000</td></tr>
<tr><td>7</td><td>VideoEnhancement1</td><td>0x1D_8000</td><td>0x2000</td><td>0x1D_A000</td></tr>
</table>
// Command executed from context image during context restore
Example: MI_LOAD_REGISTER_IMMEDIATE, ADD_CS_MMIO_START_OFFSET: true, Data:0xABCD, Register Address: 0x1C_2030
The above command when executed on RenderCS will result in a write to MMIO offset 0x2030 (0x00_2000 + 0x030) instead to 0x1C_2030. Note that</td></tr>
</table></td></tr>
</table>

RenderCS MMIO start offset is 0x2000.Table below shows the result of this command executed by various command streamers during context restore.

S.No	Command Streamer	Command Streamer MMIO Base	LRI Register Offset	LRI Written Address
1	RenderCS	0x2000	0x1C_2030	0x00_2030
2	VideoCS0	0x1C_0000	0x1C_2030	0x1C_0030
3	VideoCS1	0x1C_4000	0x1C_2030	0x1C_4030
4	VideoEnhancement0	0x1C_8000	0x1C_2030	0x1C_8030
5	VideoCS2	0x1D_0000	0x1C_2030	0x1D_0030
6	VideoCS3	0x1D_4000	0x1C_2030	0x1D_4030
7	VideoEnhancement1	0x1D_8000	0x1C_2030	0x1D_8030

0	**[Default]**	"Register Address" field in the command is absolute and not an offset from the executing command streamer MMIO start offset..

18 Reserved

Format:	MBZ

17 Reserved

Format:	MBZ

16:13 Reserved

Format:	MBZ

12 Reserved

11:8 Byte Write Disables

Range: Must specify a valid register write operation
If [11:8] is '1111b', then this command will behave as a NOOP. Otherwise, the value is forwarded to the destination register.

7:0 DWord Length

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

1..2 63:32 Data DWord

Mask:	Bytes Write Disables
Format:	U32

This field specifies the DWord value to be written to the targeted location.

31:23 Reserved

Format:	MBZ

22:2 Register Offset

MI_LOAD_REGISTER_IMM		
		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

MI_LOAD_REGISTER_MEM

MI_LOAD_REGISTER_MEM

Source: RenderCS, BlitterCS, VideoCS, VideoEnhancementCS

Length Bias: 2

The MI_LOAD_REGISTER_MEM command requests from a memory location and stores that DWord to a register.

Programming Notes

The command temporarily halts commands that will cause cycles down the 3D pipeline.

The following addresses should NOT be used for MMIO writes:

- 0x8800 - 0x88FF
- >= 0xC0000

Limited MMIO writes cycles to the Display Engine 0x40000-0xBFFFF) are allowed, but must be spaced to allow only one pending at a time. This can be done by issuing an SRM to the same address immediately after each MMIO write.

This command should not be used within a non-privilege batch buffer to access global virtual space, doing so will be treated as privilege access violation. Refer "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation.

This command is not allowed to update the privilege register range when executed from a non-privilege batch buffer.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 29h MI_LOAD_REGISTER_MEM Format: OpCode
	22	**Use Global GTT** Format: Boolean This bit if set when executing from a non-privileged batch buffer will be treated as privilege access violation. It is allowed for this bit to be clear when executing this command from a privileged (secure) batch buffer or ring buffer. This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.
	21	**Async Mode Enable** Format: Enable If this bit is set then the command stream will not wait for completion of this command before executing the next command. Please refer to the LOAD_INDIRECT and Predicate registers for

usage of this bit.

20 Reserved

19 Add CS MMIO Start Offset

This bit controls the functionality of the "Register Address" field in the command.

Value	Name	Description
1		"Register Address" field in the command is treated as an offset from the executing Command Streamer's MMIO start offset.Bits [22:2] of the "Register Address" are considered as dword offset to be added to the MMIO start offset of the corresponding command streamer. Example: MI_LOAD_REGISTER_MEM, ADD_CS_MMIO_START_OFFSET: true, Memory Address:0xABCD, Register Address: 0x1C_0030 The above command when executed on RenderCS will result in a write to MMIO offset 0x1C_2030 (0x00_2000 + 0x1C_0030) instead to 0x1C_0030. Note that RenderCS MMIO start offset is 0x2000.
0	**[Default]**	"Register Address" field in the command is absolute and not an offset from the executing command streamer MMIO start offset.

18 Reserved

Format:	MBZ

17 Reserved

Format:	MBZ

16 Reserved

Format:	MBZ

15:8 Reserved

7:0 DWord Length

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

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

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..3	63:2	**Memory Address** Format: GraphicsAddress[63:2] This field specifies the address of the memory location where the register value specified in the DWord above will read from. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[63:2] for a DWord register GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].
	1:0	**Reserved** Format: MBZ

MI_LOAD_REGISTER_REG

MI_LOAD_REGISTER_REG

Source: CommandStreamer

Length Bias: 2

The MI_LOAD_REGISTER_REG command reads from a source register location and writes that value to a destination register location.
Any offset that is to a destination outside of the GT core will allow the parser to continue once the cycle is at the GT boundary and not destination. Any other address will ensure the destination is updated prior to parsing the next command

Programming Notes

- The command temporarily halts commands that will cause cycles down the 3D pipeline.
- Destination register with mask implemented (Ex: Some registers have bits [31:16] as mask bits and bits[15:0] as data) will not get updated unless the value read from source register has the bits corresponding to the mask bits set. Note that any mask implemented register when read returns "0" for the bits corresponding to mask location. When the source and destination are mask implemented registers, destination register will not get updated with the source register contents.
- This command is not allowed to update the privilege register range when executed from a non-privilege batch buffer.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 2Ah MI_LOAD_REGISTER_REG Format: OpCode
	22:20	**Reserved** Format: MBZ
	19	**Add CS MMIO Start Offset Destination** This bit controls the functionality of the "Register Address Destination" field in the command.

Value	Name	Description
1		"Destination Register Address" field in the command is treated as an offset from the executing Command Streamer's MMIO start offset. Bits [22:2] of the "Destination Register Address" are considered as dword offset to be added to the MMIO start offset of the corresponding command streamer. Example: MI_LOAD_REGISTER_REGISTER_REG, DEST_ADD_CS_MMIO_START_OFFSET: true,

				SRC_ADD_CS_MMIO_START_OFFSET:true, Source Register Address:0x1C_0130, Destination Register Address: 0x1C_0030 The above command when executed on RenderCS will result in a MMIO read from 0x1C_2130 (0x00_2000 + 0x1C_0130) and write to MMIO offset 0x1C_2030 (0x00_2000 + 0x1C_0030) instead of read from 0x1C_0130 and write to 0x1C_0030. Note that RenderCS MMIO start offset is 0x2000.
		0	**[Default]**	"Destination Register Address" field in the command is absolute and not an offset from the executing command streamer MMIO start offset.

18 Add CS MMIO Start Offset Source

This bit controls the functionality of the "Register Address Source" field in the command.

Value	Name	Description
1		"Source Register Address" field in the command is treated as an offset from the executing Command Streamer's MMIO start offset. Bits [22:2] of the "Source Register Address" are considered as dword offset to be added to the MMIO start offset of the corresponding command streamer. Example: MI_LOAD_REGISTER_REGISTER_REG, DEST_ADD_CS_MMIO_START_OFFSET: false, SRC_ADD_CS_MMIO_START_OFFSET:true, Source Register Address:0x1C_0130, Destination Register Address: 0x1C_0030 The above command when executed on RenderCS will result in a MMIO read from 0x1C _2130 instead of read from 0x1C_0130 and write to MMIO offset 0x1C_0030. Note that RenderCS MMIO start offset is 0x2000.
0	**[Default]**	"Register Address" field in the command is absolute and not an offset from the executing command streamer MMIO start offset.

17:16 Reserved

Format:	MBZ

15:8 Reserved

Format:	MBZ

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 Source Register Address

Format:	MMIOAddress[22:2]MMIO_Register

This field specifies Bits 22:2 of the Register offset the DWord will be written to. As the register address must be DWord-aligned, Bits 1:0 of that address MBZ.

MI_LOAD_REGISTER_REG		
	1:0	**Reserved** Format: MBZ
2	31:23	**Reserved** Format: MBZ
	22:2	**Destination Register Address** Format: MMIOAddress[22:2]MMIO_Register This field specifies Bits 22:2 of the Register offset the DWord will be written to. As the register address must be DWord-aligned, Bits 1:0 of that address MBZ.
	1:0	**Reserved** Format: MBZ

MI_LOAD_SCAN_LINES_EXCL

MI_LOAD_SCAN_LINES_EXCL

Source: BlitterCS

Length Bias: 2

The MI_LOAD_SCAN_LINES_EXCL command is used to initialize the Scan Line Window registers for a specific Display Pipe. If the display refresh is inside this window the Display Engine signals the command parser to release the WAIT_FOR_EVENT command (i.e., the parser will wait while outside of the window). This command overrides the Scan Line Window defined by any previous MI_LOAD_SCAN_LINES_INCL or MI_LOAD_SCAN_LINES_EXCL commands targeting the specific display pipe.
Note: The two scan-line numbers are inclusive. If programmed to the same values, that single line defines the region in question.
Always place an even number of MI_LOAD_SCAN_LINES_EXCL/INCL at a time into the ring buffer. If only a single MI_LOAD_SCAN_LINES_EXCL/INCL is desired, just add a second identical MI_LOAD_SCAN_LINES_EXCL/INCL command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 13h MI_LOAD_SCAN_LINES_EXCL Format: OpCode
	22	**Reserved** Format: MBZ
	21:19	**Display Pipe Select** Format: U3 This field selects which Display Engine (pipe) this command is targeting. 0h: Display Pipe A 1h: Display Pipe B 2h, 3h: Reserved 4h: Display Pipe C 5h: Display Pipe D 6h, 7h: Reserved
	18:17	**Reserved**

MI_LOAD_SCAN_LINES_EXCL

	16:6	**Reserved** Format: MBZ
	5:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:16	**Start Scan Line Number** Format: U16 In scan lines, where scan line 0 is the first line of the display frame. This field specifies the starting scan line number of the Scan Line Window. Range: [0, Display Buffer height in lines-1]
	15:0	**End Scan Line Number** Format: U16 In scan lines, where scan line 0 is the first line of the display frame. This field specifies the ending scan line number of the Scan Line Window. Range: [0, Display Buffer height in lines-1]

MI_LOAD_SCAN_LINES_EXCL

MI_LOAD_SCAN_LINES_EXCL

Source: RenderCS

Length Bias: 2

The MI_LOAD_SCAN_LINES_EXCL command is used to initialize the Scan Line Window registers for a specific Display Pipe. If the display refresh is **inside** this window the Display Engine signals the command parser to release the WAIT_FOR_EVENT command (i.e., the parser will wait while outside). This command overrides the Scan Line Window defined by any previous MI_LOAD_SCAN_LINES_INCL or MI_LOAD_SCAN_LINES_EXCL commands targeting the specific display pipe. **Note:** The two scan-line numbers are inclusive. If programmed to the same values, that single line defines the region in question. Always place an even number of MI_LOAD_SCAN_LINES_EXCL/INCL at a time into the ring buffer. If only a single MI_LOAD_SCAN_LINES_EXCL/INCL is desired, just add a second identical MI_LOAD_SCAN_LINES_EXCL/INCL command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 13h MI_LOAD_SCAN_LINES_EXCL Format: OpCode
	22	**Reserved** Format: MBZ
	21:19	**Display Pipe Select** Format: U3 This field selects which Display Engine (pipe) this command is targeting. Value / Name: 0h Display Pipe A 1h Display Pipe B 2h Reserved 3h Reserved 4h Display Pipe C 5h Display Pipe D
	18:17	**Reserved**
	16:6	**Reserved**

MI_LOAD_SCAN_LINES_EXCL

		Format: MBZ
	5:0	**DWord Length** Default Value: 0h Format: =n Total Length - 2. Excludes DWord (0,1).
1	31:29	**Reserved** Format: MBZ
	28:16	**Start Scan Line Number** Format: U13 In scan lines, where scan line 0 is the first line of the display frame. Range: [0, Display Buffer height in lines-1] This field specifies the starting scan line number of the Scan Line Window.
	15:13	**Reserved** Format: MBZ
	12:0	**End Scan Line Number** Format: U13 In scan lines, where scan line 0 is the first line of the display frame. This field specifies the ending scan line number of the Scan Line Window. Range: [0, Display Buffer height in lines-1]

MI_LOAD_SCAN_LINES_INCL

MI_LOAD_SCAN_LINES_INCL

Source: BlitterCS

Length Bias: 2

The MI_LOAD_SCAN_LINES_INCL command is used to initialize the Scan Line Window registers for a specific Display Engine. If the display refresh is outside this window the Display Engine signals the command parser to release the WAIT_FOR_EVENT command (i.e., the parser will wait while inside of the window). This command overrides the Scan Line Window defined by any previous MI_LOAD_SCAN_LINES_INCL or MI_LOAD_SCAN_LINES_EXCL commands targeting the specific display.
Always place an even number of MI_LOAD_SCAN_LINES_EXCL/INCL at a time into the ring buffer. If only a single MI_LOAD_SCAN_LINES_EXCL/INCL is desired, just add a second identical

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 12h MI_LOAD_SCAN_LINES_INCL Format: OpCode
	22	**Reserved** Format: MBZ
	21:19	**Display Pipe Select** Format: U3 This field selects which Display Engine (pipe) this command is targeting. 0h: Display Pipe A 1h: Display Pipe B 2h, 3h: Reserved 4h: Display Pipe C 5h: Display Pipe D 6h, 7h: Reserved
	18:17	**Reserved**
	16:6	**Reserved**

MI_LOAD_SCAN_LINES_INCL

		Format: MBZ
	5:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:16	**Start Scan Line Number** Format: U16 In scan lines, where scan line 0 is the first line of the display frame. This field specifies the starting scan line number of the Scan Line Window. Range: [0, Display Buffer height in lines-1]
	15:0	**End Scan Line Number** Format: U16 In scan lines, where scan line 0 is the first line of the display frame. This field specifies the ending scan line number of the Scan Line Window. Range: [0, Display Buffer height in lines-1]

MI_LOAD_SCAN_LINES_INCL

MI_LOAD_SCAN_LINES_INCL

Source: RenderCS

Length Bias: 2

The MI_LOAD_SCAN_LINES_INCL command is used to initialize the Scan Line Window registers for a specific Display Engine. If the display refresh is **outside** this window the Display Engine signals the command parser to release the WAIT_FOR_EVENT command (i.e., the parser will wait while inside the window). This command overrides the Scan Line Window defined by any previous MI_LOAD_SCAN_LINES_INCL or MI_LOAD_SCAN_LINES_EXCL commands targeting the specific display.
Always place an even number of MI_LOAD_SCAN_LINES_EXCL/INCL at a time into the ring buffer. If only a single MI_LOAD_SCAN_LINES_EXCL/INCL is desired, just add a second identical MI_LOAD_SCAN_LINES_EXCL/INCL command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 12h MI_LOAD_SCAN_LINES_INCL Format: OpCode
	22	**Reserved** Format: MBZ
	21:19	**Display Pipe Select** Format: U3 This field selects which Display Engine (pipe) this command is targeting. 0h: Display Pipe A 1h: Display Pipe B 2h: Reserved 3h: Reserved 4h: Display Pipe C 5h: Display Pipe D
	18:17	**Reserved**
	16:6	**Reserved** Format: MBZ

MI_LOAD_SCAN_LINES_INCL

	5:0	**DWord Length** Default Value: 0h Format: =n Total Length - 2. Excludes DWord (0,1).
1	31	**Reserved** Format: MBZ
	30	**Reserved** Default Value: 1h Format: Must Be One
	29	**Reserved** Format: MBZ
	28:16	**Start Scan Line Number** Format: U13 In scan lines, where scan line 0 is the first line of the display frame. Range: [0, Display Buffer height in lines-1] This field specifies the starting scan line number of the Scan Line window.
	15:13	**Reserved** Format: MBZ
	12:0	**End Scan Line Number** Format: U13 In scan lines, where scan line 0 is the first line of the display frame. Range: [0, Display Buffer height in lines-1] This field specifies the ending scan line number of the Scan Line Window.

MI_MATH

MI_MATH

Source: BlitterCS

Length Bias: 2

The MI_MATH command allows software to send instructions to the ALU in the Command Streamer.
This command is the means by which the ALU is accessed. ALU instructions form the data payload of the MI_MATH command. An ALU instruction takes one DWord in size. The MI_MATH DWord Length is programmed based on the number of ALU instructions included, limited only by the max DWord Length supported.
When the command streamer parses an MI_MATH command, it sends the included ALU instructions to the ALU. The ALU processes any instruction in a single clock. See the ALU section for more details.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 1Ah MI_MATH Format: OpCode
	22:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Format: =n Total Length - 2. Excludes DWord (0,1).
1..n	31:0	**ALU INSTRUCTION** Format: Table Entry

MI_MATH

MI_MATH

Source: VideoCS

Length Bias: 2

The MI_MATH command allows software to send instructions to the ALU in the Command Streamer.
This command is the means by which the ALU is accessed. ALU instructions form the data payload of the MI_MATH command. An ALU instruction takes one DWord in size. The MI_MATH DWord Length is programmed based on the number of ALU instructions included, limited only by the max DWord Length supported.
When the command streamer parses an MI_MATH command, it sends the included ALU instructions to the ALU. The ALU processes any instruction in a single clock. See the ALU section for more details.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 1Ah MI_MATH Format: OpCode
	22:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Format: =n Total Length - 2. Excludes DWord (0,1).
1..n	31:0	**ALU INSTRUCTION** Format: Table Entry

MI_MATH

MI_MATH

Source: VideoEnhancementCS

Length Bias: 2

The MI_MATH command allows software to send instructions to the ALU in the Command Streamer.
This command is the means by which the ALU is accessed. ALU instructions form the data payload of the MI_MATH command. An ALU instruction takes one DWord in size. The MI_MATH DWord Length is programmed based on the number of ALU instructions included, limited only by the max DWord Length supported.
When the command streamer parses an MI_MATH command, it sends the included ALU instructions to the ALU. The ALU processes any instruction in a single clock. See the ALU section for more details.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 1Ah MI_MATH Format: OpCode
	22:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Format: =n Total Length - 2. Excludes DWord (0,1).
1..n	31:0	**ALU INSTRUCTION** Format: Table Entry

MI_MATH

MI_MATH

Source: RenderCS

Length Bias: 2

The MI_MATH command allows SW to send instruction to ALU in Render Command Streamer.
MI_MATH command is the means by which ALU can be accessed. ALU instructions form the data payload of MI_MATH command, ALU instruction is dword in size. MI_MATH Dword Length should be programmed based on the number of ALU instruction packed, max number is limited by the max Dword Length supported. When MI_MATH command is parsed by command streamer it outputs the payload dwords (ALU instructions) to the ALU. ALU takes single clock to process any given instruction. Refer to B-spec "Command Streamer (CS) ALU Programming" section in Command Streamer Programming.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 1Ah MI_MATH Format: OpCode
	22:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2. Excludes DWord (0,1). Value: [0h-255h] Name:
1..n	31:0	**ALU INSTRUCTION** Format: Table Entry

MI_NOOP

MI_NOOP

Source: VideoEnhancementCS

Length Bias: 1

The MI_NOOP command basically performs a "no operation" in the command stream and is typically used to pad the command stream (e.g., in order to pad out a batch buffer to a QWord boundary). However, there is one minor (optional) function this command can perform - a 22-bit value can be loaded into the MI NOPID register. This provides a general-purpose command stream tagging ("breadcrumb") mechanism (e.g., to provide sequencing information for a subsequent breakpoint interrupt).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 00h MI_NOOP Format: OpCode
	22	**Identification Number Register Write Enable** Format: Enable This field enables the value in the Identification Number field to be written into the MI NOPID register. If disabled, that register is unmodified - making this command an effective "no operation" function. Value / Name / Description 1 / / Write th NOP_ID Register 0 / / Do not write the NOP_ID register
	21:0	**Identification Number** Format: U22 This field contains a 22-bit number which can be written to the MI NOPID register.

MI_NOOP

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).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND
	28:23	**MI Command Opcode** Default Value: 0h MI_NOOP
	22	**Identification Number Register Write Enable** Format: Enable This field enables the value in the Identification Number field to be written into the MI NOPID register. If disabled, that register is unmodified - making this command an effective "no operation" function. Value / Name / Description 0h / Disable / Do not write the NOP_ID register. 1h / Enable / 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_NOOP

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.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND
	28:23	**MI Command Opcode** Default Value: 0h MI_NOOP
	22	**Identification Number Register Write Enable** Format: Enable This field enables the value in the Identification Number field to be written into the MI NOPID register. If disabled, that register is unmodified, making this command an effective "no operation" function. Value / Name / Description 0h / Disable / Do not write the NOP_ID register. 1h / Enable / 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_NOOP

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

MI_PREDICATE

Source: RenderCS

Length Bias: 1

Programming Notes
This command is supported by PositionCS.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 0Ch MI_PREDICATE Format: OpCode
	22:8	**Reserved** Format: MBZ
	7:6	**Load Operation** This field controls if/how the Predicate state bit is modified. 0h KEEP: The Predicate state bit is unmodified. 1h Reserved: 2h LOAD: The Predicate state bit is loaded with the combine operation result. 3h LOADINV: The Predicate state bit is loaded with the inverted combine operation result.
	5	**Reserved** Format: MBZ
	4:3	**Combine Operation** This field controls if/how the result of the compare operation is combined with the current Predicate state bit. 0h SET: The combine operation output the compare result unmodified. 1h AND: The combine operation outputs the AND of the compare result and the current Predicate state bit. 2h OR: The combine operation outputs the OR of the compare result and the current Predicate state bit. 3h XOR: The combine operation outputs the XOR of the compare result and the current

				Predicate state bit.

2	**Reserved** Format: MBZ

1:0 **Compare Operation**

This field controls how Data DWord 0 and Data DWord 1 fields are used to generate a compare operation result and possibly modify the PredicateData register.

Value	Name	Description
0h	TRUE	The compare operation outputs TRUE. The PredicateData register is unmodified.
1h	FALSE	The compare operation outputs FALSE. The PredicateData register is unmodified.
2h	SRCS_EQUAL	(MItemp0 - MItemp1) is computed and loaded into the PredicateData register. The compare operation outputs (MItemp0 == MItemp1).
3h	DELTAS_EQUAL	(MItemp0 - MItemp1) is computed and compared to the PredicateData register. If the values are equal, the compare operation outputs TRUE, otherwise it outputs FALSE. The PredicateData register is unmodified.

MI_REPORT_HEAD

MI_REPORT_HEAD

Source: VideoEnhancementCS

Length Bias: 1

The MI_REPORT_HEAD command causes the Head Pointer value of the ring buffer to be written to a cacheable (snooped) system memory location.
When the **Per-Process Virtual Address Space and Execlist Enable bit** is reset: The location written is relative to the address programmed in the Hardware Status Page Address Register. When the **Execlist Enable** is set, the head pointer will be reported to the PP HW Status Page.

Programming Notes

This command must not be executed from a Batch Buffer (Refer to the description of the HWS_PGA register).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 07h MI_REPORT_HEAD Format: OpCode
	22:0	**Reserved** Format: MBZ

MI_REPORT_HEAD

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.
When the **Execlist Enable** bit is reset:
The location written is relative to the address programmed in the Hardware Status Page Address Register.

Programming Notes

This command must not be executed from a Batch Buffer (Refer to the description of the HWS_PGA register).
When the **Execlist Disable** is clear, the head pointer will be reported to the PP HW Status Page.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 07h MI_REPORT_HEAD Format: OpCode
	22:0	**Reserved** Format: MBZ

MI_REPORT_HEAD

MI_REPORT_HEAD

Source: RenderCS

Length Bias: 1

The MI_REPORT_HEAD command causes the Head Pointer value of the active ring buffer to be written to a cacheable (snooped) system memory location. When Execlist Enable is set, the head pointer will be reported to the PP HW Status Page. The location written is relative to the address programmed in the Hardware Status Page Address Register.

Programming Notes

This command must not be executed from a Batch Buffer. (Refer to the description of the HWS_PGA register.)

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 07h MI_REPORT_HEAD Format: OpCode
	22:0	**Reserved** Format: MBZ

MI_REPORT_HEAD

MI_REPORT_HEAD

Source: VideoCS

Length Bias: 1

The MI_REPORT_HEAD command causes the Head Pointer value of the ring buffer to be written to a cacheable (snooped) system memory location. When the Per-Process Virtual Address Space and Execlist Enable bits reset: The location written is relative to the address programmed in the Hardware Status Page Address Register. When the Execlist Enable is set, the head pointer will be reported to the PP HW Status Page.

Programming Notes

This command must not be executed from a Batch Buffer (Refer to the description of the HWS_PGA register).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 07h MI_REPORT_HEAD Format: OpCode
	22:0	**Reserved** Format: MBZ

MI_REPORT_PERF_COUNT

MI_REPORT_PERF_COUNT

Source: RenderCS

Length Bias: 2

The MI_REPORT_PERF_COUNT command causes the GFX hardware to write out a snap-shot of performance counters to the address specified in this command along with constant ID field supplied and the time-stamp counter. This write is required to be treated as a cacheable write irrespective of GTT entry memory type. This command is specific to the render engine.

Programming Notes
SW is entirely responsible for managing the ID field and addresses used by such a series of commands.
GTT_SELECT must not be set to 1 (i.e. GGTT) when MI_REPORT_PERF_COUNT command is programmed in a non-privileged batch buffer. Refer to the "User Mode Privileged commands" Table in MI_BATCH_BUFFER_START command section for more details. All batch buffers in PPGTT are considered as Non-privileged.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 28h MI_REPORT_PERF_COUNT Format: OpCode
	22:6	**Reserved** Format: MBZ
	5:0	**DWord Length** Format: =n Total Length - 2 Value: 2h — Name: Excludes DWord (0,1) **[Default]**
1..2	63:6	**Memory Address** Format: GraphicsAddress[63:6] This field specifies 64B aligned GFX MEM address where the chap counter values are reported. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47] **Programming Notes** This field is ignored if "Report to OABUFFER" bit is set.

MI_REPORT_PERF_COUNT

	5	**Reserved** Format: MBZ
	4	**Core Mode Enable** Format: U1 This bit is set then the address will be offset by the Core ID:If Core ID 0, then there is no offsetIf Core ID 1, then the Memory is offset by the size of the data(64b).
	3:1	**Reserved** Format: MBZ
	0	**Use Global GTT** Format: Boolean This field when set (i.e. bit = 1) selects the GGTT for address translation. When this bit is 0 (default value), HW should use PGTT for address translation.
3	31:0	**Report ID** Format: U32 This field specifies the ID provided by SW for a given report command. It can be tracked to use different flavors of these reports based on where in command-stream they are inserted.This field is reported only when Counter Select Field is 0. **Programming Notes** If a privilege access violation occurs, the REPORT ID field in the report generated by the next legitimate MI_REPORT_PERF_COUNT will be corrupted.

MI_RS_STORE_DATA_IMM

MI_RS_STORE_DATA_IMM

Source: RenderCS

Length Bias: 2

The MI_RS_STORE_DATA_IMM command requests a write of the DWord constant supplied in the packet to the specified Memory Address.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 2Bh MI_RS_STORE_DATA_IMM Format: OpCode MI_RS_STORE_DATA_IMM
	22	**Reserved** Format: MBZ
	21	**Reserved**
	20:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 2h Excludes DWord (0,1) Format: =n Total Length - 2. Excludes DWord (0,1).
1..2	63:2	**Destination Address** Format: GraphicsAddress[63:2] This field specifies Bits 47:2 of the Address where the DWord will be stored. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47]. When render engine is PPGTT enabled this Address is translated using PPGTT, else GGTT is used for translation.
	1	**Reserved** Format: MBZ

MI_RS_STORE_DATA_IMM		
	0	**Core Mode Enable** If this bit is set then the address will be offset by the Core ID: If Core ID 0, then there is no offset If Core ID 1, then the Memory is offset by the size of the data.
3	31:0	**Data DWord 0** Format: U32 This field specifies the DWord value to be written to the targeted location.

MI_SEMAPHORE_SIGNAL

MI_SEMAPHORE_SIGNAL

Source: CommandStreamer

Length Bias: 2

Description

This command is used to signal the target engine stating the memory semaphore update occurrence to one of its contexts with **Target Context ID.** MI_SEMPHORE_SIGNAL and MI_SEMAPHORE_WAIT together replace the MI_SEMAPHORE_MBOX command. MI_ATOMIC (non-posted) command will be programmed prior to this command to update the semaphore data in memory.

Programming Notes

Below programming notes must be followed while programming MI_SEMAPHORE_SIGNAL for POCS. MI_SEMAPHOR_SIGNAL command in POCS command sequence may be used to signal RCS but not any other command streamer (VCS, VECS, BCS). Only RCS can semaphore signal POCS and no other command streamer must signal POCS.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 1Bh MI_SEMAPHORE_SIGNAL Format: OpCode
	22	**Reserved** Format: MBZ
	21	**Post-Sync Operation** Source: RenderCS

Value	Name	Description
0h	No Post Sync Operation	Command is executed as usual.
1h	Post Sync Operation	MI_SEMAPHORE_SIGNAL command is executed as a pipelined PIPE_CONTROL flush command with Semaphore Signal as post sync operation. Flush completion only guarantees the workload prior to this command is pushed till Windower unit and completion of any outstanding flushes issued prior to this command.

MI_SEMAPHORE_SIGNAL

		Programming Notes Any desired pipeline flush operation can be achieved by programming PIPE_CONTROL command prior to this command. When this bit is set Command Streamer sends a flush down the pipe and the atomic operation is saved as post sync operation. Command streamer goes on executing the following commands. Atomic operation saved as post sync operation is executed at some point later on completion of corresponding flush issued. When this bit is set atomic semaphore signal operation will be out of order with rest of the MI commands programmed in the ring buffer or batch buffer, it will be in order with respect to the post sync operations resulting due to PIPE_CONTROL command.
	21	**Reserved** Source: BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS Format: MBZ
	20:19	**Reserved** Format: MBZ
	18:15	**Target Engine Select** This field selects the target engine to which SIGNAL will be send to.

Value	Name	Notes
0h	RCS	
1h	VCS0	
2h	BCS	
3h	VECS	
4h	VCS1	
5h	POCS	RenderCS, PositionCS
5h	Reserved	BlitterCS, VideoCS, VideoEnhancementCS
6h	VCS2	
7h	VCS3	
Ch	VECS1	
Fh, 8h, 9h, Ah, Bh, Dh, Eh	Reserved	

	14:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2

MI_SEMAPHORE_SIGNAL		
1	31:0	**Target Context ID** In execlist based scheduling this field contains the Context ID corresponding to the context of the target engine that this command is signaling. Target engine waiting on MI_SEMPHORE_WAIT in signal mode will re-fetch the data from memory or comparison if its context ID is same as this signaled Context ID. When execlists are enabled, Target engine on receiving this Context ID sends message to the SHIM if it doesn't have the context with the same Context ID running. Message send to SHIM carries the Context ID which will be looked at by UC for rescheduling the signaled Context ID. Target engine waiting on MI_SEMAPHORE_WAIT in signal mode will fetch data from memory for comparison on receiving signal irrespective of the context id received.

MI_SEMAPHORE_WAIT

MI_SEMAPHORE_WAIT

Source: CommandStreamer

Length Bias: 2

Description

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

- If comparison passes, the command streamer moves to the next command.
- If comparison fails Command streamer switches out the context. Context switch can be inhibited by setting "Inhibit Synchronous Context Switch" in CTXT_SR_CTL register.
- If "Inhibit Synchronous context Switch" is enabled and comparison fails, Command Streamer evaluates the Compare Operation based on the Wait Mode until the compare operation is true or Wait is canceled by SW.
- CS generates semaphore wait interrupt to the scheduler when MI_SEMAPHORE_WAIT command is un-successful and when "Inhibit Synchronous Context Switch" is set. Scheduler can use this interrupt to preempt the context waiting on semaphore wait.

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

- If comparison passes, the command streamer moves to the next command.
- Unlike in Memory based semaphore, there is no context switch on an un-successful semaphore wait in “Register Poll” mode, however preemption is supported on unsuccessful semaphore wait in “Register Poll” mode. Semaphore wait interrupt is not generated by default on wait un-successful in “Register Poll” mode.
- Also unlike in Memory based semaphore, generation of an interrupt for a semaphore wait in "Register Poll" mode is not dependent on the value of bit "Inhibit Synchronous Context Switch" in register "CTXT_SR_CTL"
- Register Poll mode of Semaphore Wait command operation is non-privileged and will be supported from PPGTT batch buffers.
- HW will trigger Render DOP CG on semaphore wait unsuccessful by default and can be disabled if not desired by programming “Register Poll Mode Semaphore Wait Event IDLE message Disable” bit in “INSTPM” register. Note that Render DOP CG will not be triggered on register semaphore wait un-

successfull from INDIRECT_CTX pointer or BB_PER_CTX_PTR buffers.

Programming Notes

MI_SEMAPHORE_WAIT command must not be used in the middle of a tile pass on the posh pipe.

RenderCS synchronous context switch (Switch on semaphore wait or Display Wait For Event or Head Equal Tail) is optimized not to occur when PositionCS (POSH Pipe) is busy making forward progress. PostionCS indicates RenderCS as busy on a register based semaphore poll wait for event. Typical usage model for Register based semaphore ins PostionCS is for polling on a register to be updated by RenderCS.
The above said behavior can result in un-desirable inhibiting of synchronous context switch in RenderCS in following scenario:

- RenderCS is waiting on memory based semaphore due to external dependency (different context).
- PositionCS is waiting on register based poll to be satisfied by RenderCS.
- RenderCS is not able to satisfy PostionCS as it is struck on memory based semaphore.

However preemption is honored and context switch will take place in above mentioned scenario on doing load to submit queue.
Following Work Around must be implemented to avoid un-desirable inhibiting of synchronous context switch in RenderCS:

- Semaphores due to external dependencies must be implemented in both RenderCS and PositonCS. OR
- Use only Memory based semaphores in PostionCS.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 1Ch MI_SEMAPHORE_WAIT Format: OpCode
	22	**Memory Type** This bit will be ignored and treated as if clear when executing from a non-privileged batch buffer. It is allowed for this bit to be clear when executing this command from a privileged (secure) batch buffer. This bit *must* be 1 if the **Per Process GTT Enable** bit is clear. 0h – Per Process Graphics Address 1h – Global Graphics Address: This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.
	21:19	**Reserved** Format: MBZ

	18	**Reserved**
	17	**Reserved**
	16	**Register Poll Mode** This field control the semaphore wait behavior of polling from memory vs MMIO register.

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

Programming Notes
In register poll mode of operation of MI_SEMAPHORE_WAIT command, context switch is not supported on un-successful wait. "Wait Mode" must be always set to "Polling Mode" when Register Poll Mode is enabled. Preemption is supported on unsuccessful semaphore wait in "Register Poll" mode if operation.

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

Value	Name	Description
1h	Polling Mode	In this mode HW periodically reads the semaphore data from memory for comparison until it is context switched out. Periodicity will be mentioned in a SEMA_WAIT_POLL register.
0h	Signal Mode	[] In this mode HW will reacquire the semaphore data from memory on receiving SIGNAL with the same Context ID.

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

	14:12	**Compare Operation**

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

SAD = Semaphore Address Data
SDD = Semaphore Data Dword

MI_SEMAPHORE_WAIT

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

	11:10	**Reserved** Format: MBZ
	9:8	**Reserved**
	7:0	**DWord Length** Format: =n Total Length - 2

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

1	31:0	**Semaphore Data Dword** Format: U32 This Data dword is supplied by software to control execution of the command buffer. This value is used as part of the comparison to result in waiting or continuing in the command parser if enabled.
2..3	63:2	**Semaphore Address** Format: **GraphicsAddress63-2** **Register Poll Mode:** In Register Poll mode of operation, Bits 22:2 (Bits 63:23 are reserved MBZ, HW enforced) specify the MMIO offset of the register for the semaphore. **Non Register Poll Mode:** This field is the Graphics Memory Address of the 32-bit value for the semaphore. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form.

MI_SEMAPHORE_WAIT		
	1:0	**Reserved** Format: MBZ

MI_SET_CONTEXT

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. 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).
All context is saved and restored from a GGTT space.
This command does not initiate any interrupt due to context switch of any kind and does not support any workaround batch buffer or indirect context offset feature.

Programming Notes
For ring buffer mode, the first 128B(2 cache lines) of the context image are saved as zeros.
In execution list mode, this command must be preceded with a MI_ARB_ON_OFF command to disable arbitration and followed by a MI_ARB_ON_OFF command to enable arbitration. The first 320 bytes(5 cache lines) of the context image will be saved as zeros.
Arbitration Mode must be set to not allow lite restore prior to this command being executed. This bit is a field in the MI_ARB_ON_OFF command when in Execution list Mode.
This command needs to be always followed by a single MI_NOOP instruction to workaround a silicon issue.
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.
MI_SET_CONTEXT command must not be programmed for a POSH enabled context.

DWord	Bit	Description	
0	31:29	**Command Type**	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	**MI Command Opcode**	

MI_SET_CONTEXT

		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).
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	**Core Mode Enable** Format: Enable If set the Context Image will be offset based off the Core ID: If Core ID 0, no offset If Core ID 1, 36KB Offset
	3	**Resource Streamer State Save Enable** Format: Enable

		If set, the resource streamer state identified in the Logical Context Data section of the Memory Data Formats chapter is saved as part of switching away from this logical context. This bit will be stored in the associated CCID register to control the context save operation when switching away from this context (as part of a subsequent MI_SET_CONTEXT command).
	2	**Resource Streamer State Restore Enable** Format: Enable If set, the resource streamer state identified in the Logical Context Data section of the Memory Data Formats chapter is loaded (or restored) as part of switching to this logical context. This bit affects the switch (if required) to the context specified in Logical Context Address. This bit will also be stored in the associated CCID register to control a subsequent context save operation when switching to this context (as part of a subsequent ring buffer switch).
	1	**Force Restore** When switching to this logical context a comparison between Logical Context Address and the contests of the CCID register is performed. Normally, matching addresses prevent a context restore from occurring; however, when this bit is set a context restore is forced to occur. This bit cannot be set with Restore Inhibit. Note: This bit is not saved in the associated CCID register. It only affects the processing of this command.
	0	**Restore Inhibit** If set, the restore of the HW context from the logical context specified by Logical Context Address is inhibited (i.e., the existing HW context values are maintained). This bit must be used to prevent the loading of an uninitialized logical context. If clear, the context switch proceeds normally. This bit cannot be set with Force Restore. Note: This bit is not saved in the associated CCID register. It only affects the processing of this command.

MI_SET_PREDICATE

MI_SET_PREDICATE

Source: RenderCS

Length Bias: 1

Description

This command sets the Predication Check for the subsequent commands in the command buffer except for MI_SET_PREDICATE itself. Render Command Streamer NOOPs the following commands based on the PREDICATE_ENABLE from MI_SET_PREDICATE, MI_SET_PREDICATE_RESULT and MI_SET_PREDICATE_RESULT_2 status. Resource Streamer doesn't take any action of parsing MI_SET_PREDICATE, this command is similar to any other command which is not meant for resource streamer.

Executing MI_SET_PREDICATE command sets PREDICATE_ENABLE bits in MI_MODE register, MI_MODE register gets render context save restored.

Programming Notes

- MI_SET_PREDICATE predication scope must be confined within a Batch Buffer to set of commands.
- MI_SET_PREDICATE with Predicate Enable Must always have a corresponding MI_SET_PREDICATE with Predicate Disable within the same Batch Buffer.
- MI_ARB_CHK command must be programmed outside the Predication Scope of MI_SET_PREDICATE.
- MI_SET_PREDICATE Predication Scope must not involve any RC6 triggering events.

Only the following command(s) can be programmed between the MI_SET_PREDICATE command enabled for predication: 3DSTATE_URB_VS 3DSTATE_URB_HS 3DSTATE_URB_DS 3DSTATE_URB_GS 3DSTATE_PUSH_CONSTANT_ALLOC_VS 3DSTATE_PUSH_CONSTANT_ALLOC_HS 3DSTATE_PUSH_CONSTANT_ALLOC_DS 3DSTATE_PUSH_CONSTANT_ALLOC_GS 3DSTATE_PUSH_CONSTANT_ALLOC_PS MI_LOAD_REGISTER_IMM MEDIA_VFE_STATE MEDIA_OBJECT MEDIA_OBJJECT_WALKER MEDIA_INTERFACE_DESCRIPTOR_LOAD 3DSTATE_WM_HZ_OP MI_STORE_DATA_IMM

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 01h MI_SET_PREDICATE Format: OpCode
	22:4	**Reserved** Format: MBZ
	3:0	**Predicate Enable**

This field sets the predication logic in render command streamer when parsed. Predicate Disable is the default mode of operation.

Value	Name	Description
0h	NOOP Never	Predication is Disabled and RCS will process commands as usual.
1h	NOOP on Result2 clear	Following Commands will be NOOPED by RCS only if the MI_PREDICATE_RESULT_2 is clear.
2h	NOOP on Result2 set	Following Commands will be NOOPED by RCS only if the MI_PREDICATE_RESULT_2 is set.
3h	NOOP on Result clear	Following Commands will be NOOPED by RCS only if the MI_PREDICATE_RESULT is clear.
4h	NOOP on Result set	Following Commands will be NOOPED by RCS only if the MI_PREDICATE_RESULT is set.
Bh	NOOP in RenderCS	When RenderCS parses MI_SET_PREDICATE command with "Predicate Enable" set to "NOOP in RenderCS", RenderCS NOOP's all the subsequent commands parsed unconditionally until the predication is disabled/modified using MI_SET_PREDICATE command. Other command streamers (non RenderCS) on parsing MI_SET_PREDICATE command with "Predicate Enable" set to "NOOP in RenderCS" don't take any action and is equivalent to parsing MI_NOOP command.
Ch	NOOP in PositionCS	When PositionCS parses MI_SET_PREDICATE command with "Predicate Enable" set to "NOOP in PositionCS", PositionCS NOOP's all the subsequent commands parsed unconditionally until the predication is disabled/modified using MI_SET_PREDICATE command. Other command streamers (non PositionCS) on parsing MI_SET_PREDICATE command with "Predicate Enable" set to "NOOP in PositionCS" don't take any action and is equivalent to parsing MI_NOOP command.
Dh, Eh	Reserved	
5h, 6h, 7h, 8h, 9h, Ah	Reserved	
Fh	NOOP Always	Following Commands will be NOOPED by RCS unconditionally.

MI_STORE_DATA_IMM

MI_STORE_DATA_IMM

Source: CommandStreamer

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).

This command supports writing to multiple consecutive dwords or qwords memory locations from the starting address.

Programming Notes

- This command should not be used within a "non-privilege" batch buffer to access global virtual space, doing so will be treated as privilege access violation. Refer "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation. This command can be used within ring buffers and/or privilege batch buffers to access global virtual space.
- This command can be used for general software synchronization through variables in cacheable memory (i.e., where software does not need to poll un-cached memory or device registers).
- This command simply initiates the write operation with command execution proceeding normally. Although the write operation is guaranteed to complete eventually, there is no mechanism to synchronize command execution with the completion (or even initiation) of these operations.

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 and this command must be executing from a privileged (secure) batch buffer. If clear, the PPGTT will be used. It is allowed for this bit to be clear when executing this command from a privileged (secure) batch buffer. This bit must be '1' if the Per Process GTT Enable bit is clear.
	21	**Store Qword**

		Format: Boolean
		If set, this command generates Qword writes to memory, two "Data Dword" are paired to form a Qword. Number of qwords generated depends upon the number of "Data Dword" programmed in the command. If 'x' number of "Data Dwords" are programmed in this command it results in "x/2" qword writes to memory. If reset this command generates Dwords writes to memory. Number of dwords generated depends upon the number of "Data Dword" programmed in the command. If 'x' number of "Data Dwords" are programmed in this command it results in "x" dword writes to memory.
	20:13	**Reserved**
	12	**Reserved** Format: MBZ
	11	**Reserved**
	10	**Reserved**
	9:0	**DWord Length** Format: =n Total Length - 2. Excludes DWord (0,1)

Value	Name
2h	Store Dword **[Default]**
3h	Store Qword

Programming Notes	Source
DWord Length programmed must not exceed 0x3FE.	
If RS is enabled in the batch buffer, then the value of this field must not exceed 0x3F.	RenderCS

1..2	63:2	**Address** Format: **GraphicsAddress63-2** GraphicsAddress is 64-bit value [63:0], but only a portion of it is used by hardware. The uppermost bits are ignored and MBZ. This field specifies Bits 47:2 of the Address where the DWord will be stored. As the store address must be DWord-aligned, Bits 1:0 of that address MBZ. This address must be 8B aligned for a store "QW" command.
	1	**Reserved** Format: MBZ
	0	**Core Mode Enable**

MI_STORE_DATA_IMM		
		Format: U1 This bit is set then the address will be offset by the Core ID: If Core ID 0, then there is no offset If Core ID 1, then the Memory is offset by the size of the data(32b or 64b based off number of DW length).
3	31:0	**Data DWord 0** Format: U32 This field specifies the DWord value to be written to the targeted location.For a QWord write this DWord is the lower DWord of the QWord to be reported (DW 0).
4	31:0	**Data DWord 1** Format: U32 This field specifies the upper DWord value to be written to the targeted QWord location (DW 1).

MI_STORE_DATA_INDEX

MI_STORE_DATA_INDEX

Source: CommandStreamer

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	**Use Per-Process Hardware Status Page** **Description** If this bit is set, this command will index into the per-process hardware status page at offset 0K from the LRCA. If clear, the Global Hardware Status Page will be indexed.
	20:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 1h

MI_STORE_DATA_INDEX

		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. **Value** / **Name** [16, 1023]
	1:0	**Reserved** Format: MBZ
2	31:0	**Data DWord 0** Format: U32 This field specifies the DWord value to be written to the targeted location. For a QWord write this DWord is the lower DWord of the QWord to be reported (DW 0).
3	31:0	**Data DWord 1** Format: U32 This field specifies the upper DWord value to be written to the targeted QWord location (DW 1).

MI_STORE_REGISTER_MEM

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 should not be used from within a "non-privilege" batch buffer to access global virtual space. doing so will be treated as privilege access violation. Refer "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation. This command can be used within ring buffers and/or "privilege" batch buffers to access global virtual space.
- This command will cause undefined data to be written to memory if given register addresses for the PGTBL_CTL_0 or FENCE registers.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 24h MI_STORE_REGISTER_MEM Format: OpCode
	22	**Use Global GTT** Format: Boolean It is allowed for this bit to be set when executing this command from a privileged (secure) batch or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear. This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.
	21	**Reserved** Source: BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS

	Format:	MBZ

21 Predicate Enable

Source:	RenderCS, PositionCS
Format:	Boolean

If set, this command is executed (or not) depending on the current value of the MI_PREDICATE internal state bit in MMIO register MI_PREDICATE_RESULT[0]. This command is ignored only if PredicateEnable is set and value in the MMIO register MI_PREDICATE_RESULT[0] is 0.

20 Reserved

Format:	MBZ

19 Add CS MMIO Start Offset

This bit controls the functionality of the "Register Address" field in the command.

Value	Name	Description
1		"Register Address" field in the command is treated as an offset from the executing Command Streamer's MMIO start offset. Bits [22:2] of the "Register Address" are considered as dword offset to be added to the MMIO start offset of the corresponding command streamer. Example: MI_STORE_REGISTER_MEM, ADD_CS_MMIO_START_OFFSET: true, Memory Address:0xABCD, Register Address: 0x1C_0030 The above command when executed on RenderCS will result in updating the memory address with the content of the MMIO offset 0x1C_2030 (0x00_2000 + 0x1C_0030) instead to 0x1C_0030. Note that RenderCS MMIO start offset is 0x2000.
0	**[Default]**	"Register Address" field in the command is absolute and not an offset from the executing command streamer MMIO start offset.

18 Reserved

Format:	MBZ

17 Reserved

16 Reserved

Format:	MBZ

15:8 Reserved

7:0 DWord Length

Format:	=n Total Length - 2

MI_STORE_REGISTER_MEM

		Value / **Name** 2h — Excludes DWord (0,1) **[Default]**
1	31:23	**Reserved** Format: MBZ
	22:2	**Register Address** Format: MMIOAddress[22:2]MMIO_Register This field specifies Bits 22:2 of the Register offset the DWord will be read from. As the register address must be DWord-aligned, Bits 1:0 of that address MBZ. **Programming Notes** • Storing a VGA register is not permitted and will store an UNDEFINED value. • The values of PGTBL_CTL0 or any of the FENCE registers cannot be stored to memory; UNDEFINED values will be written to memory if the addresses of these registers are specified.
	1:0	**Reserved** Format: MBZ
2..3	63:2	**Memory Address** Format: GraphicsAddress[63:2]MMIO This field specifies the address of the memory location where the register value specified in the DWord above will be written. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[63:2] for a DWord register GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].
	1:0	**Reserved** Format: MBZ

MI_SUSPEND_FLUSH

MI_SUSPEND_FLUSH

Source: VideoEnhancementCS

Length Bias: 1

Description
Blocks PM Flush Requests.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 0Bh MI_SUSPEND_FLUSH Format: OpCode
	22:1	**Reserved** Format: MBZ
	0	**Suspend Flush** Format: Enable **Description** This field suspends flush due to a PM flush request.

MI_SUSPEND_FLUSH

MI_SUSPEND_FLUSH

Source: BlitterCS

Length Bias: 1

Description
Blocks PM Flush Requests.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 0Bh MI_SUSPEND_FLUSH Format: OpCode
	22:1	**Reserved** Format: MBZ
	0	**Suspend Flush** Format: Enable **Description** This field suspends flush due to a PM flush request.

MI_SUSPEND_FLUSH

MI_SUSPEND_FLUSH

Source: RenderCS

Length Bias: 1

Description
Blocks PM Flush Requests.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 0Bh MI_SUSPEND_FLUSH Format: OpCode
	22:1	**Reserved** Format: MBZ
	0	**Suspend Flush** Format: Enable **Description** This field suspends flush due to a PM flush request.

MI_SUSPEND_FLUSH

MI_SUSPEND_FLUSH

Source: VideoCS

Length Bias: 1

Description
Blocks PM Flush Requests.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 0Bh MI_SUSPEND_FLUSH Format: OpCode
	22:1	**Reserved** Format: MBZ
	0	**Suspend Flush** Format: Enable **Description** This field suspends flush due to a PM flush request.

MI_TOPOLOGY_FILTER

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_Prim_Topo_Type When non-zero, the CS will discard all 3DPRIMITIVE commands which do not match the specified 3DPrimTopologyType. When zero, no filtering is performed (normal operation).

MI_UPDATE_GTT

MI_UPDATE_GTT

Source: BSpec

Length Bias: 2

Description	Source
The MI_UPDATE_GTT command is used to update GTT page table entries in a coherent manner and at a predictable place in the command flow. A PIPE_CONTROL flush command with "CS Stall" bit set must be programmed prior to MI_UPDATE_GTT command, since work associated with preceding commands that are still in the pipeline may be referencing GTT entries that will be changed by its execution. The flush must also invalidate TLBs and read caches that may become invalid as a result of the changed GTT entries. A PIPE_CONTROL flush command with "CS Stall" bit set must be programmed post MI_UPDATE_GTT command to ensure the GGTT is updated with modified page table entries before the following workload references the modified entries. PIPE_CONTROL flush is not required if it can be guaranteed that the pipeline is free of any work that relies on changing GTT entries (such as MI_UPDATE_GTT contained in a paging DMA buffer that is doing only update/mapping activities and no rendering). MI_UPDTE_GTT command is privilege operation and will be converted to a no-op and an error flagged if it is executed from within a non-secure batch buffer. PPGTT updates cannot be done via **MI_UPDATE_GTT**, gfx driver will have to use MI_STORE_DATA_IMM for PPGTT inline updates.	RenderCS
The MI_UPDATE_GTT command is used to update GGTT page table entries in a coherent manner and at a predictable place in the command flow. A MI_FLUSH_DWORD flush command with "CS Stall" bit set must be programmed prior to MI_UPDATE_GTT command, since work associated with preceding commands that are still in the pipeline may be referencing GTT entries that will be changed by its execution. The flush must also invalidate TLBs and read caches that may become invalid as a result of the changed GTT entries. A MI_FLUSH_DWORD flush command with "CS Stall" bit set must be programmed post MI_UPDATE_GTT command to ensure the GGTT is updated with modified page table entries before the following workload references the modified entries. MI_FLUSH_DWORD flush is not required if it can be guaranteed that the pipeline is free of any work that relies on changing GTT entries (such as MI_UPDATE_GTT contained in a paging DMA buffer that is doing only update/mapping activities and no rendering). MI_UPDATE_GTT command is privilege operation and will be converted to a no-op and an error flagged if it is executed from within a non-secure batch buffer. PPGTT updates cannot be done via **MI_UPDATE_GTT**, gfx driver will have to use MI_STORE_DATA_IMM for PPGTT inline updates.	BlitterCS, VideoCS, VideoEnhancementCS

DWord	Bit	Description

MI_UPDATE_GTT

0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 23h MI_UPDATE_GTT Format: OpCode
	22:10	**Reserved** Format: MBZ
	9:0	**DWord Length** Default Value: 0h Format: =n Total Length - 2. Excludes DWord (0,1).
1	31:12	**Entry Address** Format: GraphicsAddress[31:12] This field holds the QW offset of the first table entry to be modified in GGTT.
	11:0	**Reserved** Format: MBZ
2..n	63:0	**Entry Data** Format: PageTableEntry This Dword becomes the new page table entry. See PPGTT/Global GTT Table Entries (PTEs) in Memory Interface Registers.

MI_USER_INTERRUPT

MI_USER_INTERRUPT

Source: VideoEnhancementCS

Length Bias: 1

The MI_USER_INTERRUPT command is used to generate a User Interrupt condition. The parser will continue parsing after processing this command. See User Interrupt.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 02h MI_USER_INTERRUPT Format: OpCode
	22:0	**Reserved** Format: MBZ

MI_USER_INTERRUPT

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 Format: OpCode
	28:23	**MI Command Opcode** Default Value: 02h MI_USER_INTERRUPT Format: OpCode
	22:0	**Reserved** Format: MBZ

MI_USER_INTERRUPT

MI_USER_INTERRUPT

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.

Programming Notes

Due to known HW issue, MI_USER_INTERRUPT followed by MI_STORE_DATA_IMM OR MI_STORE_REGISTER_MEM can cause deadlocks while post-sync operations of pipelined PIPE_CONTROL commands are asynchronously generating “Notify” interrupts.
One of the following SW WA’s can be temporarily applied to WA the HW issue until fixed. Note that that SW WA is not efficient and will cause performance and power degradation.

1. PIPE_CONTROL with “Notify Enable” must be used instead of MI_USER_INTERRUPT command. OR
2. PIPE_CONTROL with “Command Streamer stall Enable” set must be programmed prior to MI_USER_INTERRUPT

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 02h MI_USER_INTERRUPT Format: OpCode
	22:0	**Reserved** Format: MBZ

MI_USER_INTERRUPT

MI_USER_INTERRUPT		
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_2

MI_WAIT_FOR_EVENT_2

Source: RenderCS, BlitterCS

Length Bias: 1

The MI_WAIT_FOR_EVENT_2 command is used to pause command stream processing of an engine (render or blitter) until a specific display event occurs (V-Blank) or a specific condition (Flip Pending, Scanline Pending) exists.

- Display engine can be configured to generate periodic V-Blank event to an engine.
- An engine on executing MI_LOAD_SCANLINE_INCL/EXCL command for a display pipe, expects a corresponding scanline event response from display engine. Engine tracks the pending scanline response for each of the display pipe separately.
- An engine on executing MI_DISPLAY_FLIP command for a display plane, expects a corresponding flip done event response from display engine. Engine tracks the pending flip done response (flip pending) for each of the display plane separately. Display flip could be of type Sync Flip or Async Flip, hence engine also tracks the type of flip (Sync or Async) along with the pending flip done response for a given display plane.

Only one event or condition can be specified in the command -- specifying multiple events or conditions is UNDEFINED. The command parser will halt until the event occurs or condition exists on parsing this command. Note that if a specified condition (Pending Flip Done response or Pending Scanline response) does not exist at the time the parser executes this command, the parser proceeds, treating this command as a no-operation (Ex: Command is no-operation when parsed with Display Plane Flip Pending Wait Enable set to Display Plane-1 when there is no outstanding flip done response for Display Plane-1 in the engine).

Execution List Mode of Scheduling:

An engine on evaluating unsuccessful MI_WAIT_FOR_EVENT_2 (results in pausing command stream) triggers synchronous context switch stating the switch reason in Context Status Buffer. With exception of not triggering synchronous context switch on unsuccessful MI_WAIT_FOR_EVENT_2 due to Flip Pending on an Async flip. Note that synchronous context switch can be inhibited through programming Inhibit Synchronous Context Switch bit in CTXT_SR_CTL register or by disabling arbitration through MI_ARB_ON_OFF command around MI_WAIT_FOR_EVENT_2. When synchronous context switch is inhibited and the engine is waiting on an unsuccessful MI_WAIT_FOR_EVENT_2, a submission of new execlist will trigger preemption process switching out the context. With exception of not triggering preemption on an unsuccessful MI_WAIT_FOR_EVENT_2 due to Flip Pending on an Async flip.

Engine will always re-evaluate the wait condition for context switched out due to unsuccessful MI_WAIT_FOR_EVENT2 on resubmission of the context.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 04h MI_WAIT_FOR_EVENT_2

MI_WAIT_FOR_EVENT_2

	Format:	OpCode

22:15 Reserved

Format:	MBZ

14:12 Display Pipe Scan Line Wait Enable

Format:	Enable

This field enables a wait while a Display Pipe "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.

Value	Name
0h	No Wait
1h	Display Pipe A
2h	Display Pipe B
3h	Display Pipe C
4h	Display Pipe D
[5h,7h]	Reserved

11 Reserved

Format:	MBZ

10:8 Display Pipe Vertical Blank Wait Enable

Format:	Enable

This field enables a wait until the next Display Pipe "Vertical Blank" event occurs. This event is described as the start of the next Display A vertical blank period. Note that this can cause a wait for up to an entire refresh period.

Value	Name
0h	No Wait
1h	Display Pipe A
2h	Display Pipe B
3h	Display Pipe C
4h	Display Pipe D
[5h,7h]	Reserved

7:6 Reserved

Format:	MBZ

5:0 Display Plane Flip Pending Wait Enable

Format:	Enable

This field enables a wait for the duration of a Display Plane 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).

Value	Name

0h	No Wait
1h	Display Plane-1
2h	Display Plane-2
3h	Display Plane-3
4h	Display Plane-4
5h	Display Plane-5
6h	Display Plane-6
7h	Display Plane-7
8h	Display Plane-8
9h	Display Plane-9
Ah	Display Plane-10
Bh	Display Plane-11
Ch	Display Plane-12
Dh	Display Plane-13
Eh	Display Plane-14
Fh	Display Plane-15
10h	Display Plane-16
11h	Display Plane-17
12h	Display Plane-18
13h	Display Plane-19
14h	Display Plane-20
15h	Display Plane-21
16h	Display Plane-22
17h	Display Plane-23
18h	Display Plane-24
19h	Display Plane-25
1Ah	Display Plane-26
1Bh	Display Plane-27
1Ch	Display Plane-28
1Dh	Display Plane-29
1Eh	Display Plane-30
1Fh	Display Plane-31
20h	Display Plane-32
[21h, 3Fh]	Reserved

MI_WAIT_FOR_EVENT

MI_WAIT_FOR_EVENT

Source: RenderCS, BlitterCS

Length Bias: 1

The MI_WAIT_FOR_EVENTcommand is used to pause command stream processing of an engine (render or blitter) until a specific display event occurs (V-Blank) or a specific condition (Flip Pending, Scanline Pending) exists.

- Display engine can be configured to generate periodic V-Blank event to an engine.
- An engine on executing MI_LOAD_SCANLINE_INCL/EXCL command for a display pipe, expects a corresponding scanline event response from display engine. Engine tracks the pending scanline response for each of the display pipe separately.
- An engine on executing MI_DISPLAY_FLIP command for a display plane, expects a corresponding flip done event response from display engine. Engine tracks the pending flip done response (flip pending) for each of the display plane separately. Display flip could be of type Sync Flip or Async Flip, hence engine also tracks the type of flip (Sync or Async) along with the pending flip done response for a given display plane.

Only one event or condition can be specified in the command -- specifying multiple events or conditions is UNDEFINED. The command parser will halt until the event occurs or condition exists on parsing this command. Note that if a specified condition (Pending Flip Done response or Pending Scanline response) does not exist at the time the parser executes this command, the parser proceeds, treating this command as a no-operation (Ex: Command is no-operation when parsed with Display Plane Flip Pending Wait Enable set to Display Plane-1 when there is no outstanding flip done response for Display Plane-1 in the engine).

Execution List Mode of Scheduling:

An engine on evaluating unsuccessful MI_WAIT_FOR_EVENT(results in pausing command stream) triggers synchronous context switch stating the switch reason in Context Status Buffer. With exception of not triggering synchronous context switch on unsuccessful MI_WAIT_FOR_EVENTdue to Flip Pending on an Async flip. Note that synchronous context switch can be inhibited through programming Inhibit Synchronous Context Switch bit in CTXT_SR_CTL register or by disabling arbitration through MI_ARB_ON_OFF command around MI_WAIT_FOR_EVENT. When synchronous context switch is inhibited and the engine is waiting on an unsuccessful MI_WAIT_FOR_EVENT, a submission of new execlist will trigger preemption process switching out the context. With exception of not triggering preemption on an unsuccessful MI_WAIT_FOR_EVENTdue to Flip Pending on an Async flip.

Engine will always re-evaluate the wait condition for context switched out due to unsuccessful MI_WAIT_FOR_EVENTon resubmission of the context.

DWord	Bit	Description	
0	31:29	**Command Type**	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	**MI Command Opcode**	
		Default Value:	03h MI_WAIT_FOR_EVENT

	Format:	OpCode
22	**Reserved**	
	Format:	MBZ
21	**Display Plane 1 C Vertical Blank Wait Enable**	
	Format:	Enable
	This field enables a wait until the next Display Plane 1 C "Vertical Blank" event occurs. This event is described as the start of the next Display C vertical blank period. Note that this can cause a wait for up to an entire refresh period. See Vertical Blank Event in the Device Programming Interface chapter of MI Functions.	
20	**Display Plane 6 Flip Pending Wait Enable**	
	Format:	Enable
	This field enables a wait for the duration of a Display Plane 2 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	**Display Plane 12 Flip Pending Wait Enable**	
	Format:	Enable
	This field enables a wait for the duration of a Display Plane 4 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).	
18	**Display Plane 11 Flip Pending Wait Enable**	
	Format:	Enable
	This field enables a wait for the duration of a Display Plane 4 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).	
17	**Display Plane 10 Flip Pending Wait Enable**	
	Format:	Enable
	This field enables a wait for the duration of a Display Plane 4 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).	
16	**Display Plane 9 Flip Pending Wait Enable**	
	Format:	Enable
	This field enables a wait for the duration of a Display Plane 3 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).	
15	**Display Plane 3 Flip Pending Wait Enable**	

		Format: Enable This field enables a wait for the duration of a Display Plane 1 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 Plane 1 C Scan Line Wait Enable** Format: Enable This field enables a wait while a Display Plane 1 C "Scan Line" condition exists. This condition is defined as the the start of the scan line specified in the Pipe C Display Scan Line Count Range Compare Register.
	13:12	**Reserved** Format: MBZ
	11	**Display Plane 1 B Vertical Blank Wait Enable** Format: Enable This field enables a wait until the next Display Plane 1 B "Vertical Blank" event occurs. This event is described as the start of the next Display B vertical blank period. Note that this can cause a wait for up to an entire refresh period.
	10	**Display Plane 5 Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 2 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 2 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 Plane 1 B Scan Line Wait Enable** Format: Enable This field enables a wait while a Display Plane 1 B "Scan Line" condition exists. This condition is defined as the the start of the scan line specified in the Pipe B Display Scan Line Count Range Compare Register.
	7	**Display Plane 8 Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 3 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).
	6	**Display Plane 7 Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 3 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).
	5:4	**Reserved** Format: MBZ
	3	**Display Plane 1 A Vertical Blank Wait Enable** Format: Enable This field enables a wait until the next Display Plane 1 A "Vertical Blank" event occurs. This event is described as the start of the next Display A vertical blank period. Note that this can cause a wait for up to an entire refresh period.
	2	**Display Plane 4 Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 2 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 1 Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 1 A Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).
	0	**Display Plane 1 A Scan Line Wait Enable** Format: Enable This field enables a wait while a Display Plane 1 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.

Monitor Event MSD

MSD_MONITOR_EVENT - Monitor Event MSD		
Source:	EuSubFunctionGateway	
Length Bias:	1	
Gateway will remember if this Event ID occurs.		
DWord	**Bit**	**Description**
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Default Value: 0b Format: Enable **Restriction** Must be zero.
	18:3	**Reserved** Format: MBZ
	2:0	**Monitor Event Subfunction** Default Value: 010b Format: OpCode

Monitor No Event MSD

MSD_MONITOR_NO_EVENT - Monitor No Event MSD		
Source:	EuSubFunctionGateway	
Length Bias:	1	
Gateway will no longer record any Events for sending thread.		
DWord	**Bit**	**Description**
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Default Value: 0b Format: Enable **Restriction** Must be zero.
	18:3	**Reserved** Format: MBZ
	2:0	**Monitor No Event Subfunction** Default Value: 011b Format: OpCode

Move

mov - Move

Source:	Eulsa
Length Bias:	4

The mov instruction moves the components in src0 into the channels of dst. If src0 and dst are of different types, format conversion is performed. If src0 is a scalar immediate, the immediate value is loaded into enabled channels of dst. A mov with the same source and destination type, no source modifier, and no saturation is a raw move. A packed byte destination region (B or UB type with HorzStride == 1 and ExecSize > 1) can only be written using raw move.

When denorm mode is flush to zero, a raw mov instruction with saturation modifier will not flush the denorm input or output to zero (Denorm is preserved).

Format:

```
[(pred)] mov[.cmod] (exec_size) dst src0
```

Programming Notes

A *mov* instruction with a source modifier always copies a denorm source value to a denorm destination value (in the manner of a raw move).

There is no direct conversion from B/UB to DF or DF to B/UB. Use two instructions and a word or DWord intermediate type.

There is no direct conversion from B/UB to Q/UQ or Q/UQ to B/UB. Use two instructions and a word or DWord intermediate integer type.

There is no direct conversion from HF to DF or DF to HF. Use two instructions and F (Float) as an intermediate type.

There is no direct conversion from HF to Q/UQ or Q/UQ to HF. Use two instructions and F (Float) or a word integer type or a DWord integer type as an intermediate type.

Restriction

Raw move is not supported for Float values in ALT mode if any values are infinities or NaNs.

An accumulator can be a source or destination operand but not both.

Syntax

```
[(pred)] mov[.cmod] (exec_size) reg reg
         [(pred)] mov[.cmod] (exec_size) reg imm32
         [(pred)] mov[.cmod] (exec_size) reg imm64
```

Pseudocode

```
Evaluate(WrEn);
         for ( n = 0; n < exec_size; n++ ) {
            if ( WrEn.chan[n] ) {
               dst.chan[n] = src0.chan[n];
            }
```

mov - Move

```
}
```

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	*W,*D
*B,*W,*D	HF
F	HF
HF	*B,*W,*D
HF	F
HF	HF

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

Move Indexed

movi - Move Indexed

Source:	Eulsa
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
```

Programming Notes

HW Implementation Details:

The source register is calculated by adding the register portion of the first index register with the register portion of the address immediate, a0.0[11:5] + addr_imm[9:5]

For byte movi, byte0 of the destination is selected by (a0.0[4:0]), byte1 is selected by (a0.1[4:0]), ..., and byte7 is selected by (a0.7[4:0]). The rest of the bytes are undefined.

For word movi, byte0 of the destination is selected by (a0.0[4:1] & 0), byte1 is selected by (a0.0[4:1] & 1), byte2 is selected by (a0.1[4:1] & 0), byte3 is selected by (a0.1[4:1] & 1), ..., and byte15 is selected by (a0.7[4:1] & 1). The rest of the bytes are undefined.

For DWord or float movi, byte0 of the destination is selected by (a0.0[4:2] & 00b), byte1 is selected by (a0.0[4:2] & 01b), byte2 is selected by (a0.0[4:2] & 10b), byte3 is selected by (a0.0[4:2] & 11b), byte4 is selected by (a0.1[4:2] & 00b), byte5 is selected by (a0.1[4:2] & 01b), ..., byte31 is selected by (a0.7[4:2] & 11b).

For all 3 conditions above, a0.n[4:0] = a0.n[4:0] + addr_imm[4:0].

Restriction

- Source operand cannot be accumulators. The source operand must be a general register.
- The source and destination must have the same type.
- The address register for the source must be a0.0 or a0.8.
- The destination register (directly or indirectly addressed) must be 16-byte aligned.
- The destination region (directly or indirectly addressed) must point to the same GRF register.
- The destination stride in bytes must equal the source element size in bytes.
- The Align16 access mode is not allowed.
- All the index registers (address subregisters) used must point to the same GRF register.
- The instruction must use 1x1 indirect regioning.

movi - Move Indexed

The destination offset is only used to create channel enables. Each element of the destination is directly mapped to the index registers for the movi instruction. i.e. a0.0 -> dst.0, a0.1 -> dst.1, a0.2 -> dst.2, etc.

Only 8 address subregisters are used (a0.0-a0.7 or a0.8-a0.15). Destination element will be sourced from address register (a0.0 or a0.8), for example:

movi (8) r31.0:uw r[a0.0,0]<8;8,1>:uw // r31.0:uw<-a0.0:uw, r31.1:uw<-a0.1:uw, etc.
movi (8) r31.0:uw r[a0.8,0]<8;8,1>:uw // r31.0:uw<-a0.8:uw, r31.1:uw<-a0.9:uw, etc.
movi (8) r31.8:uw r[a0.0,0]<8;8,1>:uw // r31.8:uw<-a0.0:uw, r31.9:uw<-a0.1:uw, etc.
movi (8) r31.8:uw r[a0.8,0]<8;8,1>:uw // r31.8:uw<-a0.8:uw, r31.9:uw<-a0.9:uw, etc.
movi (8) r31.0:ud r[a0.0,0]<8;8,1>:ud // r31.0:ud<-a0.0:ud, r31.1:ud<-a0.1:ud, etc.
movi (8) r31.0:ud r[a0.8,0]<8;8,1>:ud // r31.0:ud<-a0.8:ud, r31.1:ud<-a0.9:ud, etc.

Conditional Modifier is not allowed for this instruction.

Syntax

```
[(pred)] movi (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
             srcregfile = regfile(src0);
             imm_offset = (src1 == NULL) ? addr_imm : src1;
             srcregbase = reg(address[0]) + reg(imm_offset);
             for ( n = 0; n < RegWidth; n++ ) {
                 if ( WrEn.chan[n] ) {
                     srcsubreg = subreg(address[n] + imm_offset);
                     dst.chan[n] = srcregfile.srcreg.srcsubreg;
                 }
             }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	Y	Y

Src Types	Dst Types
B	B
UB	UB
W	W
UW	UW
D	D
UD	UD
F	F

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG

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

Multiply

mul - Multiply

Source: Eulsa

Length Bias: 4

The mul instruction performs component-wise multiplication of src0 and src1 and stores the results in dst.
When multiplying integer datatypes, if src0 is DW and src1 is W, irrespective of the destination datatype, the accumulator maintains full 48-bit precision. This is required to handle the macro for 32x32 multiplication. The macro described in the mach instruction should be used to obtain the full precision 64-bit multiplication results.
Note: A 32x32 multiply operation is handled natively, without a macro. When operating in this mode, the resulting 64-bit data is packed, unlike the macro, where the lower and upper 32 bits of the result are written to different general registers by two separate instructions. Refer to the macro description for details.
When multiplying integer data types, if one of the sources is a DW, the resulting full precision data is stored in the accumulator. However, if the destination data type is either W or DW, the low bits of the result are written to the destination register and the remaining high bits are discarded. This results in undefined Overflow and Sign flags. Therefore, conditional modifiers and saturation (.sat) cannot be used in this case.

Format:

```
[(pred)] mul[.cmod] (exec_size) dst src0 src1
```

Restriction

Integer source operands cannot be accumulators.

When multiplying a DW and any lower precision integer, the DW operand must on src0.

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

mul - Multiply

*W	*W
*W	*D
F	F
HF	HF
HF, F	HF, F

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

Multiply Accumulate

mac - Multiply Accumulate

Source: Eulsa

Length Bias: 4

The mac instruction takes component-wise multiplication of src0 and src1, adds the results with the corresponding accumulator values, and then stores the final results in dst.

Format:

```
[(pred)] mac[.cmod] (exec_size) dst src0 src1
```

Programming Notes

When source and destination datatypes are different, the implied datatype for the accumulator operand is always the destination datatype.

Restriction

Accumulator is an implicit source and thus cannot be an explicit source operand.

The conditional modifier and saturation (.sat) must not be used when src0 or src1 are dwords.

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
HF	HF
HF, F	HF, F

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG_REG

mac - Multiply Accumulate

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

Multiply Accumulate High

mach - Multiply Accumulate High

Source: Eulsa

Length Bias: 4

The mach instruction performs DWord integer multiply-accumulate operation and outputs the high DWord (bits 63:32). For each enabled channel, this instruction multiplies the DWord in src0 with the high word of the DWord in src1, left shifts the result by 16 bits, adds it with the corresponding accumulator values, and keeps the whole 64-bit result in the accumulator. It then stores the high DWord (bits 63:32) of the results in dst. This instruction is intended to be used to emulate 32-bit DWord integer multiplication by using the large number of bits available in the accumulator. For example, the following instructions perform vector multiplication of two 32-bit signed integer sources from r2 and r3 and store the resulting vectors with the high 32 bits in r5 and the low 32 bits in r6.

```
mul (8) acc0:d r2.0<8;8,1>:d r3.0<16;8,2>:uw
mach (8) r5.0<1>:d r2.0<8;8,1>:d r3.0<8;8,1>:d
mov (8) r6.0<1>:d acc0:d // Low 32 bits.
```

Here is a different example including negation. An added preliminary mov is required for source modification on src1.

```
mov (8) r3.0<1>:d -r3<8;8,1>:d
mul (8) acc0:d r2.0<8;8,1>:d r3.0<16;8,2>:uw
mach (8) r5.0<1>:d r2.0<8;8,1>:d r3.0<8;8,1>:d // High 32 bits
mov (8) r6.0<1>:d acc0:d // Low 32 bits.
```

The mach should have channel enable from the destHI of IMUL, the mov should have the channel enable from the destLO of IMUL. As mach is used to generate part of the 64-bit DWord integer results, saturation modifier should not be used. In fact, saturation modifier should not be used for any of these four instructions. Source and destination operands must be DWord integers. Source and destination must be of the same type, signed integer or unsigned integer. If dst is UD, src0 and src1 may be UD and/or D. However, if any of src0 and src1 is D, source modifier (abs) must be present to convert it to match with dst. If dst is D, src0 and src1 must also be D. They cannot be UD as it may cause unexpected overflow because the computed results are limited to 64 bits.

Format:

```
          [(pred)] mach[.cmod] (exec_size) dst src0 src1
```

Restriction

Accumulator is an implicit source and thus cannot be an explicit source operand.

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
```

Pseudocode

```
Evaluate(WrEn);
   for ( n = 0; n < exec size; n++ ) {
```

mach - Multiply Accumulate High

```
      if ( WrEn.chan[n] ) {
         temp.chan[n][63:0] = (src1.chan[n][31:16] *
            src0.chan[n][31:0]) « 16 + acc.chan[n][63:0];
         if (AccWrEn) {
            acc.chan[n][63:0] = temp.chan[n][63:0];
            dst.chan[n][31:0] = temp.chan[n][63:32];
         }
         else {
            dst.chan[n][31:0] = temp.chan[n][31:0];
         }
       }
    }
```

Errata	Description
	A source modifier must not be used on src1 for the macro operation. This applies to both mul and mach of the macro. If source modifier is required, an additional mov instruction may be used before the macro.

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	Y	Y

Src Types	Dst Types
D	D
UD	UD

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

Multiply Add

mad - Multiply Add

Source: Eulsa

Length Bias: 4

Description

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.
The conditional modifier and saturation (.sat) must not be used when src1 or src2 are dwords.

```
Plane and Linear Interpolation instructions are removed. The following macros must be used
to emulate Plane and Linear Interpolation operations.

 Plane Instruction Emulation

 The below plane instruction

 pln (16) r20.0<1>:f r10.4<0;1,0>:f r4.0<8;8,1>:f

 is emulated as below

 mad (8) acc0<1>:nf r10.7<0;1,0>:f r4.0<8;8,1>:f r10.4<0;1,0>:f
 mad (8) r20.0<1>:f acc0<8;8,1>:nf r5.0<8;8,1>:f r10.5<0;1,0>:f
 mad (8) acc0<1>:nf r10.7<0;1,0>:f r6.0<8;8,1>:f r10.4<0;1,0>:f
 mad (8) r21.0<1>:f acc0<8;8,1>:nf r7.0<8;8,1>:f r10.5<0;1,0>:f

 In case of SIMD8 pln instruction only the first pair of mad instructions are used.

 Linear Interpolation Instruction Emulation

 The below lrp instruction

 lrp (16) r40.0<1>:f r10.0<8;8,1>:f r20.0<8;8,1>:f r30.0<8;8,1>:f

 is emulated as below

 mad (8) acc0<1>:nf r30.0<8;8,1>:f r10.0<8;8,1>:f r20.0<8;8,1>:f
 mad (8) r40.0<1>:f acc0<8;8,1>:nf -r10.0<8;8,1>:f r30.0<8;8,1>:f
 mad (8) acc0<1>:nf r31.0<8;8,1>:f r11.0<8;8,1>:f r21.0<8;8,1>:f
 mad (8) r41.0<1>:f acc0<8;8,1>:nf -r11.0<8;8,1>:f r31.0<8;8,1>:f

 In case of SIMD8 lrp instruction only the first pair of mad instructions are used.
```

Format:

```
[(pred)] mad[.cmod] (exec_size) dst src0 src1 src2
```

Restriction

Src1/Src2 for Integer source operands cannot be accumulators. Src0 is allowed to use accumulator.

All three-source instructions have certain restrictions, described in Instruction Formats.

Syntax

mad - Multiply Add

```
[(pred)] mad[.cmod] (exec_size) reg  reg   reg reg
         [(pred)] mad[.cmod] (exec_size) reg  reg   reg imm16
         [(pred)] mad[.cmod] (exec_size) reg  imm16 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
HF	HF
HF, F	HF, F
B	W
W, D	W, D
F, NF	F, NF

DWord	Bit	Description
0..3	127	**Reserved** Format: MBZ
	126:106	**Source 2** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC
	105:85	**Source 1** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC
	84:64	**Source 0** Format: EU_INSTRUCTION_OPERAND_SRC_REG_THREE_SRC
	63:56	**Destination Register Number** Format: DstRegNum
	55:53	**Destination Subregister Number**
	52:49	**Destination Channel Enable** Format: ChanEn[4] Four channel enables are defined for controlling which channels are written into the

destination region. These channel mask bits are applied in a modulo-four manner to all ExecSize channels. There is 1-bit Channel Enable for each channel within the group of 4. If the bit is cleared, the write for the corresponding channel is disabled. If the bit is set, the write is enabled. Mnemonics for the bit being set for the group of 4 are "x", "y", "z", and "w", respectively, where "x" corresponds to Channel 0 in the group and "w" corresponds to channel 3 in the group

48:46 **Destination Data Type**

Value	Name	Description
000b	:f	single precision Float (32-bit)
001b	:d	signed Doubleword integer
010b	:ud	Unsigned Doubleword integer
011b	:df	Double precision Float (64-bit)
100b	:hf	Half Float (16-bit)
101b-111b	Reserved	

45:43 **Source Data Type**

Value	Name	Description
000b	:f	single precision Float (32-bit)
001b	:d	signed Doubleword integer
010b	:ud	Unsigned Doubleword integer
011b	:df	Double precision Float (64-bit)
100b	:hf	Half Float (16-bit)
101b-111b	Reserved	

42:41 **Source 2 Modifier**

Exists If:	(Property[Source Modifier]=='true')
Format:	**SrcMod**

42:37 **Reserved**

Exists If:	(Property[Source Modifier]=='false')
Format:	MBZ

40:39 **Source 1 Modifier**

Exists If:	(Property[Source Modifier]=='true')
Format:	**SrcMod**

38:37 **Source 0 Modifier**

Exists If:	(Property[Source Modifier]=='true')
Format:	**SrcMod**

36 **Source 1 Type**

mad - Multiply Add

		Format: U1 Only used if Source Data Type is :f or :hf, else Source 1 Data Type matches Source 0 type and this bit is ignored. **Value / Name / Description** 0b / :f / single precision Float (32-bit) 1b / :hf / Half Float (16-bit)
	35	**Source 2 Type** Format: U1 Only used if Source Data Type is :f or :hf, else Source 2 Data Type matches Source 0 type and this bit is ignored. **Value / Name / Description** 0b / :f / single precision Float (32-bit) 1b / :hf / Half Float (16-bit)
	34	**MaskCtrl** (Formerly WECtrl/Write Enable Control). This flag disables the normal write enables; it should normally be 0. **Value / Name / Description** 0 / Normal / Use the normal write enables in Dst.ChanEn (normal setting). 1 / NoMask / Write all channels except those disabled by predication or by other masks besides the write enables. **Programming Notes** MaskCtrl = NoMask also skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
	33	**Flag Register Number** This field contains the flag register number for instructions with a non-zero Conditional Modifier.
	32	**Flag Subregister Number** This field contains the flag subregister number for instructions with a non-zero Conditional Modifier.
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

No Operation

nop - No Operation

Source: Eulsa

Length Bias: 4

Do nothing. The nop instruction takes an instruction dispatch but performs no operation. It can be used for assembly patching in memory, or to insert a delay in the program sequence.

Format:

```
nop
```

Restriction

The nop instruction takes no instruction options other than Breakpoint.

Syntax

nop

Pseudocode

```
{ ; // The null statement, which does nothing. }
```

Predication	Conditional Modifier	Saturation	Source Modifier
N	N	N	N

DWord	Bit	Description
0..3	127:31	**Reserved** Format: MBZ
	30	**Reserved**
	29:7	**Reserved** Format: MBZ
	6:0	**Opcode** Format: EU_OPCODE

Oword Aligned Block Read MSD

MSD0R_OWAB - Oword Aligned Block Read MSD

Source: EuSubFunctionDataPort0
Length Bias: 1
Family: Block R/W
Group: OW Aligned Block R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18	**Legacy Message** Default Value: 0h Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 01h Format: Opcode Aligned Block Read message

MSD0R_OWAB - Oword Aligned Block Read MSD

	13	**Reserved** Format: MBZ Ignored
	12:11	**Reserved** Format: MBZ Ignored
	10:8	**Data Elements** Format: MDC_DB_OW Specifies the number of contiguous Owords to be read
	7:0	**Binding Table Index** Format: MDC_BTS_A32 Specifies the Binding Table Index for the message

Oword Block Read MSD

MSD0R_OWB - Oword Block Read MSD

Source:	EuSubFunctionDataPort0
Length Bias:	1
Family:	Block R/W
Group:	OW Block R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18	**Legacy Message** Default Value: 0h Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 00h Format: Opcode Block Read message
	13	**Block Message Subtype** Default Value: 0

MSD0R_OWB - Oword Block Read MSD

		Format: Opcode Oword Block Read/Write subtype
	12:11	**Reserved** Format: MBZ Ignored
	10:8	**Data Elements** Format: MDC_DB_OW Specifies the number of contiguous Owords to be read or written
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies the Binding Table Index for the message

Oword Block Write MSD

MSD0W_OWB - Oword Block Write MSD

Source:	EuSubFunctionDataPort0
Length Bias:	1
Family:	Block R/W
Group:	OW Block R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18	**Legacy Message** Default Value: 0h Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 08h Format: Opcode Block Write message
	13	**Block Message Subtype** Default Value: 0 Format: Opcode

MSD0W_OWB - Oword Block Write MSD

		Oword Block subtype
	12:11	**Reserved** Format: MBZ Ignored
	10:8	**Data Elements** Format: MDC_DB_OW Specifies the number of contiguous Owords to be read or written
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies the Binding Table Index for the message

PIPE_CONTROL

PIPE_CONTROL

Source: RenderCS

Length Bias: 2

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

Programming Notes

SW must follow below programming restrictions when programming PIPECONTROL command for POCS:

- Write cache flush bits must not be set (Render Target Cache Flush Enable, DC Flush Enable, Depth Cache Flush Enable)
- Post Sync Operations must not be set to "Write PS Depth Count"
- "Stall at Pixel Scoreboard" must not be set
- "Notify Enable" must not be set.
- "Depth Stall Enable" must not be set.
- "Generic Media State Clear" must not be set.
- "PSD Sync Enable" must not be set.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 2h PIPE_CONTROL Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 0h PIPE_CONTROL Format: OpCode
	15	**Reserved** Format: MBZ
	14	**Reserved**
	13	**Reserved**

	12	**Reserved**
	11	**Reserved** Format: MBZ
	10	**Reserved** Format: MBZ
	9	**Reserved**
	8	**Reserved**
	7:0	**DWord Length** Default Value: 4h DWORD_COUNT_n Format: =n Total Length - 2. Excludes DWord (0,1).
1	31	**Reserved** Format: MBZ
	30	**Reserved** Format: MBZ
	29	**Command Cache Invalidate Enable** Format: Enable When set the command cache for commands parsed at the top of the pipe will be invalidated. This bit is independent from the other bits in this command and will be executed prior to the pipeline being flushed.
	28	**Tile Cache Flush Enable** Setting this bit will force Tile Cache (contains both color and depth data) to be flushed to memory prior to this synchronization point completing. This bit must be set for all write fence sync operations to assure that results from operations initiated prior to this command are visible in memory once software observes this synchronization.

Value	Name	Description
0		Tile Cache is not flushed.
1		Tile cache is flushed.

Programming Notes

- SW must always set CS Stall bit when Tile Cache Flush Enable bit is set in the PIPECONTROL command.
- SW must ensure level1 depth and color caches are flushed prior to flushing the tile cache. This can be achieved by following means:

- Single PIPECONTROL command to flush level1 caches and the tile cache. Attributes listed below must be set. OR
  - Tile Cache Flush Enable
  - Render Target Cache Flush Enable
  - Depth Cache Flush Enable
- Flushing of L1 caches followed by flushing of tile cache through two different PIEPCONTROL commands. SW must ensure not to issue any rendering commands between the two PIPECONTROL commands.

27 **Reserved**

26 **Flush LLC**

Format:	Enable

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

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

25 **Reserved**

Format:	MBZ

24 **Destination Address Type**

Defines address space of Destination Address

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

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

23 **LRI Post Sync Operation**

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

Programming Notes
This bit causes a post sync operation with an LRI (Load Register Immediate) operation. If this

bit is set then the Post-Sync Operation field must be cleared.

22 **Reserved**

21 **Store Data Index**

Format:	U1

Description
This field is valid only if the post-sync operation is not 0. If this bit is set, the store data address is index into the global hardware status page when destination address type in the command is set to 1 (GGTT). The store data address is index into the per-process hardware status page when destination address type in the command is set to 0 (PPGTT).

20 **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.

19 **Global Snapshot Count Reset**

Format:	U1

Value	Name	Description
0h	Don't Reset	Do not reset the snapshot counts or Statistics Counters.
1h	Reset	Reset the snapshot count in Gen4 for all the units and reset the Statistics Counters except as noted above.

Programming Notes
TIMESTAMP is not reset by PIPE_CONTROL with this bit set. When Post Sync Operation is set to "Write PS Depth Count" along with Global Snapshot Count Reset, PS Depth Count is Reported first before resetting the value.

18 **TLB Invalidate**

Format:	U1

If ENABLED, all TLBs belonging to Render Engine will be invalidated once the flush operation is complete. Note that if the flush TLB invalidation mode is clear, a TLB invalidate will occur irrespective of this bit setting
If ENABLED, PIPE_CONTROL command will flush the in flight data written out by render engine to Global Observation point on flush done. Also Requires stall bit ([20] of DW1) set.

Programming Notes

If ENABLED, all TLBs belonging to Render Engine will be invalidated once the flush operation is complete. Note that if the flush TLB invalidation mode is clear, a TLB invalidate will occur irrespective of this bit setting.

Post Sync Operation or CS stall must be set to ensure a TLB invalidate occurs. Otherwise no cycle will occur to the TLB cache to invalidate.

17 **PSD Sync Enable**

Format:	Enable

If set, Pixel Shader Dispatch Units will stall successive PS threads from being dispatched until all the prior PS threads complete. Once all PSDs are synced up, post-sync LRI can be optionally used to change EU enables.

Programming Notes

If this bit is set, these bits must not be set:

- Depth Stall Enable
- Command Streamer Stall Enable
- Render Target Cache Flush Enable
- Stall At Pixel Scoreboard

16 **Generic Media State Clear**

Format:	Disable

If set, all generic media state context information will be invalidated. Any state invalidated will not be saved as part of the render engine context image. The state only becomes valid once it is parsed by the command streamer.

15:14 **Post Sync Operation**

Description

This field specifies an optional action to be taken upon completion of the synchronization operation.

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

Value	Name	Description	Programming Notes
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	Write the 64-bit	[] Workaround

	Depth Count	PS_DEPTH_COUNT register to the Destination Address	Driver must program PIPE_CONTROL with only Depth Stall Enable bit set prior to programming a PIPE_CONTROL with Write PS Depth Count Post sync operation.
3h	Write Timestamp	Write the 64-bit TIMESTAMP register(i.e. "Reported Timestamp Count" 0x2358 for render pipe) to the Destination Address.	

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

13 **Depth Stall Enable**

Format:	Enable

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

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

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

12 **Render Target Cache Flush Enable**

Format:	Enable

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

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

		Programming Notes Whenever a Binding Table Index (BTI) used by a Render Target Message points to a different RENDER_SURFACE_STATE, SW must issue a Render Target Cache Flush by enabling this bit. When render target flush is set due to new association of BTI, PS Scoreboard Stall bit must be set in this packet.
	11	**Instruction Cache Invalidate Enable** Format: Enable Setting this bit is independent of any other bit in this packet. This bit controls the invalidation of the L1 and L2 at the top of the pipe i.e. at the parsing time.
	10	**Texture Cache Invalidation Enable** Format: Enable Setting this bit is independent of any other bit in this packet. This bit controls the invalidation of the texture caches at the top of the pipe i.e. at the parsing time. **Workaround** Workaround: "CS Stall" bit in PIPE_CONTROL command must be always set for GPGPU workloads when "Texture Cache Invalidation Enable" bit is set
	9	**Indirect State Pointers Disable** Format: Enable **Description** At the completion of the post-sync operation associated with this pipe control packet, the indirect state pointers in the hardware are considered invalid; the indirect pointers are not saved in the context. If any new indirect state commands are executed in the command stream while the pipe control is pending, the new indirect state commands are preserved. Using Invalidate State Pointer (ISP) only inhibits context restoring of Push Constant (3DSTATE_CONSTANT_*) commands. Push Constant commands are only considered as Indirect State Pointers. Once ISP is issued in a context, SW must initialize by programming push constant commands for all the shaders (at least to zero length) before attempting any rendering operation for the same context. **Programming Notes** When executed in POCS results in inhibiting the context restore of 3DSTATE_CONSTANT_VS from POSH engine context.
	8	**Notify Enable**

		Format: Enable If ENABLED, a Sync Completion Interrupt will be generated (if enabled by the MI Interrupt Control registers) once the sync operation is complete. See Interrupt Control Registers in Memory Interface Registers for details.
	7	**Pipe Control Flush Enable** Format: Enable Hardware on parsing PIPECONTROL command with Pipe Control Flush Enable set will wait for all the outstanding post sync operations corresponding to previously executed PIPECONTROL commands are complete before making forward progress.
	6	**Reserved**
	5	**DC Flush Enable** Format: Enable Setting this bit enables flushing of the L3$ portions that caches DC writes. **Programming Notes** DC Flush (L3 Flush) by default doesn't result in flushing/invalidating the IA Coherent lines from L3$, however this can be achieved by setting control bit "**Pipe line flush Coherent lines"** in "L3SQCREG4" register.
	4	**VF Cache Invalidation Enable** Format: Enable Setting this bit is independent of any other bit in this packet. This bit controls the invalidation of VF address based cache at the top of the pipe i.e. at the parsing time. **Programming Notes** When executed from POCS it results in invalidating the VFR cache.
	3	**Constant Cache Invalidation Enable** Format: Enable Setting this bit is independent of any other bit in this packet. This bit controls the invalidation of the constant cache at the top of the pipe i.e. at the parsing time.
	2	**State Cache Invalidation Enable** Format: Enable Setting this bit is independent of any other bit in this packet. This bit controls the invalidation of the L1 and L2 state caches at the top of the pipe i.e. at the parsing time. **Programming Notes** When "State Cache redirect to CS Section enable" bit is set in MMIO register

		SLICE_COMMON_ECO_CHCICKEN1 (0731Ch), “Command Cache Invalidate Enable” must be also set upon setting “State Cache Invalidate” bit in PIPE_CONTROL command.
	1	**Stall At Pixel Scoreboard** Format: Enable Defines the behavior of PIPE_CONTROL command at the pixel scoreboard. **Value** / **Name** / **Description** 0h / Disable / Stall at the pixel scoreboard is disabled. 1h / Enable / Stall at the pixel scoreboard is enabled. **Programming Notes** This bit must be DISABLED for End-of-pipe (Read) fences, PS_DEPTH_COUNT or TIMESTAMP queries. This bit is ignored if Depth Stall Enable is set. Further the render cache is not flushed even if Write Cache Flush Enable bit is set.
	0	**Depth Cache Flush Enable** Format: Enable Setting this bit enables flushing (i.e. writing back the dirty lines to memory and invalidating the tags) of depth related caches. This bit applies to HiZ cache, Stencil cache and depth cache. **Value** / **Name** / **Description** 0h / Flush Disabled / Depth relates caches (HiZ, Stencil and Depth) are NOT flushed. 1h / Flush Enabled / Depth relates caches (HiZ, Stencil and Depth) are flushed. **Programming Notes** Ideally depth caches need to be flushed only when depth is required to be coherent in memory for later use as a texture, source or honoring CPU lock. This bit must be DISABLED for End-of-pipe (Read) fences, PS_DEPTH_COUNT or TIMESTAMP queries.
2	31:2	**Address** Format: GraphicsAddress[31:2]U32 If **Post Sync Operation** is set to 1h. **LRI Post-Sync Operation** must be clear): Bits 31:3 specify 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 **LRI Post-Sync Operation** is set: Bits 22:2 (Bits 31:23 are reserved MBZ) specify the MMIO offset destination for the data in the **Immediate Data Low** (DW3) field. Only DW writes are valid.
	1:0	**Reserved** Format: MBZ
3	31:0	**Address High**

		Format: GraphicsAddress[63:32]U32 This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space. This field is valid only if the post-sync operation is not 0 and the LRI Post-Sync Operation is clear.
4..5	63:0	**Immediate Data** Format: U64 This field specifies the QWord value to be written to the targeted location. Only valid when Post-Sync Operation is 1h (Write Immediate Data) or LRI Post-Sync Operation is set. Ignored if Post-Sync Operation is "No write", "Write PS_DEPTH_COUNT" or "Write TIMESTAMP".

PIPELINE_SELECT

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 Format: OpCode
	28:27	**Command SubType** Default Value: 1h GFXPIPE_SINGLE_DW Format: OpCode
	26:24	**3D Command Opcode** Format: OpCode Value 1h: GFXPIPE_NONPIPELINED **[Default]**
	23:16	**3D Command Sub Opcode** Default Value: 04h PIPELINE_SELECT Format: OpCode
	15:8	**Mask Bits**

		Programming Notes Must be set to modify corresponding bits in Bits 7:0. (For implemented bits)
	7	**Reserved**
	6	**Media Sampler Power Clock Gate Disable** Format: U1 By default, the media power clock gating is always ON. When set, Command Streamer sends message to PM to disable media sampler power Clock Gating. **Programming Notes** Mask bit [14] has to be set for HW to look at this field when PIPELINE_SELECT command is parsed. Each BB/workload is responsible to set this control. This can be only enabled/disabled at the frame level.
	5	**Force Media Awake** Format: Enable **Value** / **Name** / **Description** 0 / Disabled / Command streamer sends message to PM to disable force awake of media engine (next instructions do not require the media engine to be awake). Command streamer waits for acknowledge from PM before parsing the next command. 1 / Enabled / Command streamer sends message to PM to force awake media engine (next instructions require media engine awake). Command streamer waits for acknowledge from PM before parsing the next command. **Programming Notes** Mask bit [13] has to be set for HW to look at this field when PIPELINE_SELECT command is parsed. Example for usage model: **RCS Ring Buffer**: **PIPELINE_SELECT (Force Media Awake set to '1')** MI_SEMPAHORE_SINGAL (Signal context id 0xABC to Render Command Streamer) **PIPELINE_SELECT (Force Media Awake set to '0')MI_BATCH_BUFFER_START** STATE Commands .. .. **PIPELINE_SELECT (Force Media Awake set to '1')** MI_LOAD_REGISTER_IMM (Load register 0x23XX in render command streamer with data 0xFFF) **PIPELINE_SELECT (Force Media Awake set to '0')** ..... **MI_BATCH_BUFFER_END**
	4	**Media Sampler DOP Clock Gate Enable** Format: Enable **Value** / **Name** / **Description** 0 / Disabled / Command Streamer sends message to PM to disable sampler DOP Clock Gating. 1 / Enabled / Command Streamer sends message to PM to enable media sampler DOP

		Clock Gating.

Programming Notes
Mask bit [12] has to be set for HW to look at this field when PIPELINE_SELECT command is parsed.

3 **Render Sampler Power Gate Enable**

Format:	Enable

Value	Name	Description
0	Disabled	Command Streamer sends message to PM to disable render sampler Power Gating.
1	Enabled	Command Streamer sends message to PM to enable render sampler Power Gating.

Programming Notes
Mask bit [11] has to be set for HW to look at this field when PIPELINE_SELECT command is parsed.

2 **Render Slice common Power Gate Enable**

Format:	Enable

Value	Name	Description
0	Disabled	Command Streamer sends message to PM to disable render slice common Power Gating.
1	Enabled	Command Streamer sends message to PM to enable render slice common Power Gating.

Programming Notes
Mask bit [10] has to be set for HW to look at this field when PIPELINE_SELECT command is parsed.

1:0 **Pipeline Selection**

Value	Name	Description
0	3D	3D pipeline is selected
1	Media	Media pipeline is selected (Includes HD optical disc playback, HD video playback, and generic media workloads)
2	GPGPU	GPGPU pipeline is selected

Programming Notes
Mask bits [9:8] has to be set for HW to look at this field when PIPELINE_SELECT command is parsed. Setting only one of the mask bit [9] or [8] is illegal.

Read Surface Info MSD

MSD_RSI - Read Surface Info MSD

Source: EuSubFunctionReadOnlyDataPort
Length Bias: 1
Family: Other
Group: Read Surface Info

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF Indicates that the message forbids a header.
	18	**Reserved** Format: MBZ Ignored
	17:14	**Message Type** Default Value: 06h

MSD_RSI - Read Surface Info MSD

		Format: Opcode Read Surface Info message
	13:8	**Reserved** Format: MBZ Ignored
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

REP16 Render Target Write MSD

MSD_RTW_REP16 - REP16 Render Target Write MSD

Source: EuSubFunctionRenderDataPort
Length Bias: 1
Family: Other
Group: Render Target R/W

DWord	Bit	Description
0	31	**Reserved** Format: MBZ Ignored
	30	**Message Precision Subtype** Default Value: 0h Format: Opcode Full precision data message
	29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present**

	Format:	MDC_MHP

If set, indicates that the message includes the 2-register header.

18 Per-Coarse Pixel PS outputs enable

Format:	Enable

This bit indicates the render target write is a coarse pixel write.

Programming Notes

This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.

17:14 Message Type

Default Value:	0Ch
Format:	Opcode

Render Target Write message

13 Per-Sample PS Enable

Format:	Enable

If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot.

Programming Notes

This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE.
When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0.
When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples).
When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.

12 Last Render Target Select

Format:	Enable

MSD_RTW_REP16 - REP16 Render Target Write MSD

		This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.
	11	**Slot Group Select** Format: MDC_RT_SGS This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
	10:8	**Render Target Message Subtype** Default Value: 1h Format: Opcode SIMD16 Single source message with replicated data. Use slots [15:0] for pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [31:16] are referenced instead of [15:0].
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Reserved Instruction0

Reserved Instruction0		
Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Format: Opcode
	28:27	**Opcode 1** Format: Opcode
	26:24	**Opcode 2** Format: Opcode
	23:16	**Opcode 3** Format: Opcode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction1

Reserved Instruction1		
Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Format: Opcode
	28:27	**Opcode 1** Format: Opcode
	26:24	**Opcode 2** Format: Opcode
	23:21	**Opcode 3** Format: Opcode
	20:16	**Opcode 4** Format: Opcode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction2

Reserved Instruction2		
Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Format: Opcode
	28:27	**Opcode 1** Format: Opcode
	26:23	**Opcode 2** Format: Opcode
	22:21	**Opcode 3** Format: Opcode
	20:16	**Opcode 4** Format: Opcode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction3

Reserved Instruction3		
Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Format: Opcode
	28:27	**Opcode 1** Format: Opcode
	26:23	**Opcode 2** Format: Opcode
	22:21	**Opcode 3** Format: Opcode
	20:16	**Opcode 4** Format: Opcode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction4

Reserved Instruction4		
Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Format: Opcode
	28:27	**Opcode 1** Format: Opcode
	26:23	**Opcode 2** Format: Opcode
	22:21	**Opcode 3** Format: Opcode
	20:16	**Opcode 4** Format: Opcode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction5

Reserved Instruction5		
Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Format: Opcode
	28:27	**Opcode 1** Format: Opcode
	26:23	**Opcode 2** Format: Opcode
	22:16	**Opcode 3** Format: Opcode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction6

Reserved Instruction6		
Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Format: Opcode
	28:27	**Opcode 1** Format: Opcode
	26:23	**Opcode 2** Format: Opcode
	22:16	**Opcode 3** Format: Opcode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction7

Reserved Instruction7		
Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Format: Opcode
	28:27	**Opcode 1** Format: Opcode
	26:23	**Opcode 2** Format: Opcode
	22:16	**Opcode 3** Format: Opcode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction8

Reserved Instruction8		
Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Format: Opcode
	28:27	**Opcode 1** Format: Opcode
	26:23	**Opcode 2** Format: Opcode
	22:16	**Opcode 3** Format: Opcode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction9

Reserved Instruction9		
Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Format: Opcode
	28:27	**Opcode 1** Format: Opcode
	26:23	**Opcode 2** Format: Opcode
	22:16	**Opcode 3** Format: Opcode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction10

Reserved Instruction10		
Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Format: Opcode
	28:27	**Opcode 1** Format: Opcode
	26:23	**Opcode 2** Format: Opcode
	22:16	**Opcode 3** Format: Opcode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction11

Reserved Instruction11		
Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Format: Opcode
	28:27	**Opcode 1** Format: Opcode
	26:23	**Opcode 2** Format: Opcode
	22:16	**Opcode 3** Format: Opcode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction12

Reserved Instruction12		
Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Format: Opcode
	28:27	**Opcode 1** Format: Opcode
	26:23	**Opcode 2** Format: Opcode
	22:16	**Opcode 3** Format: Opcode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction13

Reserved Instruction13		
Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Format: Opcode
	28:27	**Opcode 1** Format: Opcode
	26:23	**Opcode 2** Format: Opcode
	22:16	**Opcode 3** Format: Opcode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction14

Reserved Instruction14		
Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Format: Opcode
	28:27	**Opcode 1** Format: Opcode
	26:23	**Opcode 2** Format: Opcode
	22:16	**Opcode 3** Format: Opcode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction15

Reserved Instruction15		
Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Format: Opcode
	28:27	**Opcode 1** Format: Opcode
	26:23	**Opcode 2** Format: Opcode
	22:16	**Opcode 3** Format: Opcode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction16

Reserved Instruction16		
Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Format: Opcode
	28:27	**Opcode 1** Format: Opcode
	26:23	**Opcode 2** Format: Opcode
	22:16	**Opcode 3** Format: Opcode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction17

Reserved Instruction17		
Length Bias: 1		
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Format: Opcode
	28:23	**Opcode 1** Format: Opcode
	22:0	**Reserved** Format: MBZ
0..n	31:0	**Unknown Bitfield**

Reserved Instruction18

Reserved Instruction18		
Length Bias: 1		
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Format: Opcode
	28:23	**Opcode 1** Format: Opcode
	22:0	**Reserved** Format: MBZ
0..n	31:0	**Unknown Bitfield**

Reserved Instruction19

Reserved Instruction19		
Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Format: Opcode
	28:16	**Opcode 1** Format: Opcode
	15:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction20

Reserved Instruction20		
Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Format: Opcode
	28:16	**Opcode 1** Format: Opcode
	15:0	**Reserved** Format: MBZ
0..n	31:0	**Unknown Bitfield**

Reserved Instruction21

Reserved Instruction21		
Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Format: Opcode
	28:16	**Opcode 1** Format: Opcode
	15:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction22

Reserved Instruction22		
Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Format: Opcode
	28:16	**Opcode 1** Format: Opcode
	15:0	**Reserved** Format: MBZ
0..n	31:0	**Unknown Bitfield**

Return

ret - Return

Source: Eulsa

Length Bias: 4

Description

Return execution to the code sequence that called a subroutine. The ret instruction can be predicated or non-predicated. If non-predicated, all channels jump to the return IP in the first channel of src0 and restore CallMask from the second channel of src0. If predicated, the enabled channels jump to the return IP from the first channel of src0 and the corresponding bits in the CallMask are cleared to zero; if all CallMask bits are zero after the ret instruction, then execution jumps to the return IP from the first channel of src0. When SPF is on, the predication control must be scalar.

Format:

```
[(pred)] ret (exec_size) null src0
```

Restriction

This instruction cannot take accumulator as source.

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

ret - Return

D, UD

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

Rotate Left

rol - Rotate Left

Source: Eulsa

Length Bias: 4

Perform component-wise logical rotate left operation of the bits in src0 by the rotate count indicated in src1, storing the result in dst. src0 and src1 are treated as unsigned numbers with only the bits within the specified datatype used during this operation. This operation does not produce sign or overflow conditions. Only the .e/.z or .ne/.nz conditional modifiers are supported. Extra precision bits available in accumulator are ignored during this operation and only the bits within the specified datatype are used.
src0 and dst must be of same datatype precision.

Format:

```
[(pred)] rol[.cmod] (exec_size) dst src0 src1
```

Syntax

```
[(pred)] rol[.cmod] (exec_size) reg reg reg
         [(pred)] rol[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         RotateMask = sizeof(src0) * 8 - 1;
         dst.chan[n] = (src0.chan[n] « (src1.chan[n] & RotateMask)) |
                       (src0.chan[n] » (-src1.chan[n] & RotateMask));
     }
 }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	N	N

Src Types	Dst Types
UW, UD	UW, UD

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

rol - Rotate Left		
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

Rotate Right

ror - Rotate Right

Source: Eulsa

Length Bias: 4

Perform component-wise logical rotate right operation of the bits in src0 by the rotate count indicated in src1, storing the result in dst. src0 and src1 are treated as unsigned numbers with only the bits within the specified datatype used during this operation. This operation does not produce sign or overflow conditions. Only the .e/.z or .ne/.nz conditional modifiers are supported. Extra precision bits available in accumulator are ignored during this operation and only the bits within the specified datatype are used.
src0 and dst must be of same datatype precision.

Format:

```
[(pred)] ror[.cmod] (exec_size) dst src0 src1
```

Syntax

```
[(pred)] ror[.cmod] (exec_size) reg reg reg
         [(pred)] ror[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         RotateMask = sizeof(src0) * 8 - 1;
         dst.chan[n] = (src0.chan[n] » (src1.chan[n] & RotateMask)) |
                       (src0.chan[n] « (-src1.chan[n] & RotateMask));
     }
 }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	N	N

Src Types	Dst Types
UW, UD	UW, UD

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

ror - Rotate Right		
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

Round Down

rndd - Round Down

Source: Eulsa

Length Bias: 4

The rndd instruction takes component-wise floating point downward rounding (to the integral float number closer to negative infinity) of src0 and storing the rounded integral float results in dst. This is commonly referred to as the floor() function. Each result follows the rules in the following tables based on the floating-point mode.

Format:

```
[(pred)] rndd[.cmod] (exec_size) dst src0
```

Syntax

```
[(pred)] rndd[.cmod] (exec_size) reg reg
[(pred)] rndd[.cmod] (exec_size) reg imm32
```

Pseudocode

```
Evaluate(WrEn);
for ( n = 0; n < exec_size; n++ ) {
   if ( WrEn.chan[n] ) {
      dst.chan[n] = floor(src0.chan[n]);
   }
}
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types
F	F

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

Round to Nearest or Even

rnde - Round to Nearest or Even

Source: Eulsa

Length Bias: 4

The rnde instruction takes component-wise floating point round-to-even operation of src0 with results in two pieces - a downward rounded integral float results stored in dst and the round-to-even increments stored in the rounding increment bits. The round-to-even increment must be added to the results in dst to create the final round-to-even values to emulate the round-to-even operation, commonly known as the round() function. The final results are the one of the two integral float values that is nearer to the input values. If the neither possibility is nearer, the even alternative is chosen. Each result follows the rules in the following tables based on the floating-point mode.

Format:

```
[(pred)] rnde[.cmod] (exec_size) dst src0
```

Syntax

```
[(pred)] rnde[.cmod] (exec_size) reg reg
[(pred)] rnde[.cmod] (exec_size) reg imm32
```

Pseudocode

```
Evaluate(WrEn);
for ( n = 0; n < exec_size; n++ ) {
   if ( WrEn.chan[n] ) {
      if ( src0.chan[n] - floor(src0.chan[n]) > 0.5f ) {
         dst.chan[n] = floor(src0.chan[n]) + 1;
      }
      else if ( src0.chan[n] - floor(src0.chan[n]) < 0.5f ) {
         dst.chan[n] = floor(src0.chan[n]);
      }
      else {
         if ( floor(src0.chan[n]) is odd ) {
            dst.chan[n] = floor(src0.chan[n]) + 1;
         }
         else {
            dst.chan[n] = floor(src0.chan[n]);
         }
      }
   }
}
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types
F	F

DWord	Bit	Description

rnde - Round to Nearest or Even		
0..3	127:64	**RegSource** Exists If: ([Operand Controls][Src0.RegFile]!='IMM') Format: EU_INSTRUCTION_SOURCES_REG
	127:64	**ImmSource** Exists If: ([Operand Controls][Src0.RegFile]=='IMM') Format: EU_INSTRUCTION_SOURCES_IMM32
	63:32	**Operand Controls** Format: EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header** Format: EU_INSTRUCTION_HEADER

Round to Zero

rndz - Round to Zero

Source: Eulsa

Length Bias: 4

The rndz instruction takes component-wise floating point round-to-zero operation of src0 with results in two pieces - a downward rounded integral float results stored in dst and the round-to-zero increments stored in the rounding increment bits. The round-to-zero increment must be added to the results in dst to create the final round-to-zero values to emulate the round-to-zero operation, commonly known as the truncate() function. The final results are the one of the two closest integral float values to the input values that is nearer to zero.

Format:

```
[(pred)] rndz[.cmod] (exec_size) dst src0
```

Syntax

```
[(pred)] rndz[.cmod] (exec_size) reg reg
[(pred)] rndz[.cmod] (exec_size) reg imm32
```

Pseudocode

```
Evaluate(WrEn);
for ( n = 0; n < exec_size; n++ ) {
   if ( WrEn.chan[n] ) {
      dst.chan[n] = floor(src0.chan[n]);
      if ( abs(src0.chan[n]) < abs(dst.chan[n]) ) {
         dst.chan[n] = floor(src0.chan[n]) + 1;
      }
      else {
         dst.chan[n] = floor(src0.chan[n]);
      }
   }
}
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types
F	F

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

rndz - Round to Zero			
		Format:	EU_INSTRUCTION_SOURCES_IMM32
	63:32	**Operand Controls**	
		Format:	EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header**	
		Format:	EU_INSTRUCTION_HEADER

Round Up

rndu - Round Up

Source: Eulsa

Length Bias: 4

The rndu instruction takes component-wise floating point upward rounding (to the integral float number closer to positive infinity) of src0, commonly known as the ceiling() function. Each result follows the rules in the following tables based on the floating-point mode.

Format:

```
[(pred)] rndu[.cmod] (exec_size) dst src0
```

Syntax

```
[(pred)] rndu[.cmod] (exec_size) reg reg
[(pred)] rndu[.cmod] (exec_size) reg imm32
```

Pseudocode

```
Evaluate(WrEn);
for ( n = 0; n < exec_size; n++ ) {
   if ( WrEn.chan[n] ) {
      if ( src0.chan[n] - floor(src0.chan[n]) > 0.0f ) {
         dst.chan[n] = floor(src0.chan[n]) + 1;
      }
      else {
         dst.chan[n] = src0.chan[n];
      }
   }
}
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types
F	F

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

rndu - Round Up			
		Format:	EU_INSTRUCTION_HEADER

Sampler Cache Media Block Read MSD

MSD_SC_MB - Sampler Cache Media Block Read MSD

Source: EuSubFunctionReadOnlyDataPort
Length Bias: 1
Family: Other
Group: Media Block R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18	**Reserved** Format: MBZ Ignored
	17:14	**Message Type** Default Value: 05h

MSD_SC_MB - Sampler Cache Media Block Read MSD

		Format: Opcode Media Block Read Sampler Cache message
	13:11	**Reserved** Format: MBZ Ignored
	10:8	**Vertical Line Stride Override** Format: MDC_VLSO If enabled, specifies the Vertical Line Stride and Vertical Line Stride Offset override fields.
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Sampler Cache Oword Aligned Block Read MSD

MSD_SC_OWAB - Sampler Cache Oword Aligned Block Read MSD

Source: EuSubFunctionReadOnlyDataPort
Length Bias: 1
Family: Block R/W
Group: OW Aligned Block R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18	**Reserved** Format: MBZ Ignored
	17:14	**Message Type** Default Value: 04h

MSD_SC_OWAB - Sampler Cache Oword Aligned Block Read MSD

		Format: Opcode Oword Aligned Block Read Sampler Cache message
	13:11	**Reserved** Format: MBZ Ignored
	10:8	**Data Elements** Format: MDC_DB_OW Specifies the number of contiguous Owords to be read
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Scaled Untyped Surface Read MSD

MSD2R_US - Scaled Untyped Surface Read MSD

Source: EuSubFunctionDataPort2
Length Bias: 1
Family: Untyped Surface R/W
Group: Scaled Untyped Surface R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_A32_MHP If present, modifies the address calculations.
	18:15	**Message Type** Default Value: 01h Format: Opcode Untyped Surface Read message
	14	**Reserved**

MSD2R_US - Scaled Untyped Surface Read MSD

		Format: MBZ Ignored
	13:12	**SIMD Mode** Format: **MDC_SM3** Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Channel Mask** Format: **MDC_CMASK** Specifies which RGBA channels are included in the message payload.
	7:0	**Sideband Scaled Offset** Format: **MDC_A32_SBSO** In combination with Header Present field, specifies the Scale pitch and the Offset for the message.

Scaled Untyped Surface Write MSD

MSD2W_US - Scaled Untyped Surface Write MSD

Source: EuSubFunctionDataPort2
Length Bias: 1
Family: Untyped Surface R/W
Group: Scaled Untyped Surface R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_A32_MHP If present, modifies the address calculations.
	18:15	**Message Type** Default Value: 09h Format: Opcode Untyped Surface Write message
	14	**Reserved**

MSD2W_US - Scaled Untyped Surface Write MSD

		Format: MBZ Ignored
	13:12	**SIMD Mode** Format: **MDC_SM3** Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Channel Mask** Format: **MDC_UW_CMASK** Specifies which RGBA channels are included in the message payload.
	7:0	**Sideband Scaled Offset** Format: **MDC_A32_SBSO** In combination with Header Present field, specifies the Scale pitch and the Offset for the message.

Scattered Move

smov - Scattered Move

Source: Eulsa

Length Bias: 4

The smov instruction moves the components in src0 into dst. For each enabled channel, copy src0 to dst. The immediate is used to selectively enable channels without using flags. When predication is enabled, the predicate mask is not generated from the flags. Instead, the immediate is used to mask the execution mask. If any channel is enabled as a result of this masking, the instruction is executed. When predication is not enabled, the immediate masks the execution mask. This provides flexibility to mask out any channel with an immediate.

Format:

```
[(pred)] smov[.cmod] (exec_size) dst src0 src1
```

Programming Notes

When predication is disabled, the immediate provides the flexibility to perform a select operation without the use of flags.

When predication is enabled, the usage model provides flexibility to select any bit in the flag registers for predication for execution size of 1.

Syntax

```
[(pred)] smov[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
if pred emask = OR (emask AND imm32) Else pmask = imm32. Evaluate(WrEn); for ( n = 0; n < 32; n++ ) { if ( WrEn.chan[n] ) { dst.chan[n] = src0.chan[n]; } }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

Src Types	Dst Types
*W,*D, HF, F	*W,*D, HF, F

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

smov - Scattered Move			
		Format:	EU_INSTRUCTION_OPERAND_CONTROLS
	31:0	**Header**	
		Format:	EU_INSTRUCTION_HEADER

Scratch Block Read MSD

MSD0R_SB - Scratch Block Read MSD

Source: EuSubFunctionDataPort0
Length Bias: 1
Family: Block R/W
Group: HW Block R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18	**Scratch Block Message** Default Value: 1h Format: Opcode Scratch Block Message
	17	**Operation Type** Default Value: 0h

MSD0R_SB - Scratch Block Read MSD

		Format: Opcode
		Scratch Block Read message
	16	**Reserved** Format: MBZ Ignored
	15	**Invalidate After Read** Format: MDC_IAR Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	14	**Reserved** Format: MBZ Ignored
	13:12	**Data Elements** Format: MDC_DB_HW Specifies the number of registers to be read or written
	11:0	**Address Offset** Format: GeneralStateOffset[16:5] HWORD (32 byte) based address offset to the BufferAddress in the Message Header.

Scratch Block Write MSD

MSD0W_SB - Scratch Block Write MSD

Source: EuSubFunctionDataPort0
Length Bias: 1
Family: Block R/W
Group: HW Block R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18	**Scratch Block Message** Default Value: 1h Format: Opcode Scratch Block Message
	17	**Operation Type** Default Value: 1h

MSD0W_SB - Scratch Block Write MSD

		Format: Opcode
		Scratch Block Write message
	16	**Reserved** Format: MBZ Ignored
	15:14	**Reserved** Format: MBZ Ignored
	13:12	**Data Elements** Format: MDC_DB_HW Specifies the number of registers to be read or written
	11:0	**Address Offset** Format: GeneralStateOffset[17:6] HWORD (32 byte) based address offset to the BufferAddress in the Message Header.

Select

sel - Select

Source: Eulsa

Length Bias: 4

Description

The sel instruction selectively moves the components in src0 or src1 into the channels of dst based on the predication. On a channel by channel basis, if the channel condition is true, data in src0 is moved into dst. Otherwise, data in src1 is moved into dst.

As the predication is used to select the two sources, it is not included in the evaluation of WrEn. The predicate clause is mandatory if cmod is omitted/0000b. If both predication and the conditional modifier are omitted, the results are undefined.

If the conditional modifier is specified (not 0000b, a compare is performed and the resulting condition flag is used for the sel instruction. Conditional modifiers .ge and .l follow the cmpn rules, and all other conditional modifiers follow the cmp rules. Predication is not allowed in this mode.

A sel instruction with cmod .l is used to emulate a MIN instruction.

A sel instruction with cmod .ge is used to emulate a MAX instruction.

For a sel instruction with a .l or .ge conditional modifier, if one source is NaN and the other not NaN, the non-NaN source is the result. If both sources are NaNs, the result is NaN. For all other conditional modifiers, if either source is NaN then src1 is selected.

A sel instruction without a conditional modifier always copies a denorm source value to a denorm destination value (in the manner of a raw move). This applies even if the source modifies are set on the *sel* instruction sources.

The sel instruction uses any conditional modifier internally and does not update the flag register if a conditional modifier is used.

A sel instruction with cmod or source modifier will flush denorm to zero, depending on the denorm mode bit; a sel instruction without cmod and source modifier will retain denorm.

Format:

```
(pred) sel[.cmod] (exec_size) dst src0 src1
```

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];
```

sel - Select

```
                }
             }
          }
       }
       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];
                }
             }
          }
       }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

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

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

Send Message

send - Send Message

Source: Eulsa

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 message descriptor field <ex_desc> contains the target function ID. WrEn is forwarded to the target function in the message sideband.
The extended message descriptor field <ex_desc> also contains the extended function control field to be sent to the Target Shared Function over 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 32-bit immediate, imm32. The lower 4bits of the <ex_desc> specifies the SFID for the message. The bit5 of the extended message descriptor, the EOT field, always comes from bit 127 of the instruction word. A thread must terminate with a send instruction with EOT turned on. The higher 16bits, bit31:16 specify the 16bit extended function control field. Interpretation of the extended function control signals is subject to the target external function.
<src> is a 256-bit aligned GRF register. It serves as the leading GRF register of the request.
The source dependency control, {NoSrcDepSet} is used to control the setting of source dependency for the source.
<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,

send - Send Message

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 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. 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

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 encodings Switch is 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. Any instruction updating the ARF must use a {Switch} if the ARF is not used before EOT. DepCtrl Must not be used with Send Instruction. When pagefault is enabled, the source and destination operands must not overlap. This is required to ensure the messages can be replayed.</src>

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

Syntax

```
[(pred)] send (exec_size) reg reg imm32 reg32a
             [(pred)] send (exec_size) reg reg imm32 imm32
```

Pseudocode

```
Evaluate(WrEn);
           <MsgChEnable> = WrEn;
           <SourceReg> = <src>.RegNum;
           MessageEnqueue(<MsgChEnable>, <ResponseReg>, <SourceReg>, <desc>, <ex_dest>);
```

send - Send Message

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	**Reserved** Format: MBZ
	94:91	**ExDesc[31:28]** Format: ExtMsgDescpt[31:28]
	90:89	**Reserved** Format: MBZ
	88:85	**ExDesc[27:24]** Format: ExtMsgDescpt[27:24]
	84	**Reserved** Format: MBZ
	83:80	**ExDesc[23:20]** Format: ExtMsgDescpt[23:20]
	79:68	**Reserved** Format: MBZ
	67:64	**ExDesc[19:16]** Format: ExMsgDescpt[19:16]
	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**

send - Send Message			
		Format:	EU_INSTRUCTION_CONTROLS_A
	7	**Reserved**	
		Format:	MBZ
	6:0	**Opcode**	
		Format:	EU_OPCODE

SFC_AVS_CHROMA_Coeff_Table

SFC_AVS_CHROMA_Coeff_Table

Source: BSpec

Length Bias: 2

This command is sent from VDBOX/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:23	**Media Command Opcode** Default Value: Ah Media Misc Format: OpCode
	22:21	**SubOpcodeA** Default Value: 0h Common Format: OpCode
	20:16	**SubOpcodeB** Default Value: 6h SFC_AVS CHROMA Coeff_Table Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Total Length - 2 Value: 3Fh — Name: Excludes DWord (0,1) **[Default]**
1..64	2047:0	**AVS CHROMA Coefficient Table Body** Format: SFC_AVS_CHROMA_COEFF_TABLE_BODY

SFC_AVS_LUMA_Coeff_Table

SFC_AVS_LUMA_Coeff_Table

Source: BSpec

Length Bias: 2

This command is sent from VDBOX/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:23	**Media Command Opcode** Default Value: Ah Media Misc Format: OpCode
	22:21	**SubOpcodeA** Default Value: 0h Common Format: OpCode
	20:16	**SubOpcodeB** Default Value: 5h SFC_AVS LUMA Coeff_Table Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Total Length - 2 Value: 7Fh — Name: Excludes DWord (0,1) **[Default]**
1..132	4223:0	**AVS LUMA Coefficient Table Body** Format: SFC_AVS_LUMA_COEFF_TABLE_BODY

SFC_AVS_STATE

SFC_AVS_STATE

Source: BSpec

Length Bias: 2

This command is sent from VDBOX/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:23	**Media Command Opcode** Default Value: Ah Media Misc Format: OpCode
	22:21	**SubOpcodeA** Default Value: 0h Common Format: OpCode
	20:16	**SubOpcodeB** Default Value: 2h SFC_AVS_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Total Length - 2 Value: 2h — Name: Excludes DWord (0,1) **[Default]**
1..3	95:0	**AVS State Body** Format: SFC_AVS_STATE_BODY

SFC_FRAME_START

SFC_FRAME_START

Source: BSpec

Length Bias: 2

This command is sent from VDBOX/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:23	**Media Command Opcode** Default Value: Ah Media Misc Format: OpCode
	22:21	**SubOpcodeA** Default Value: 0h Common Format: OpCode
	20:16	**SubOpcodeB** Default Value: 4h SFC_FRAME_START Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:0	**Frame Start Body** Format: SFC_FRAME_START_BODY

SFC_IEF_STATE

SFC_IEF_STATE

Source: BSpec

Length Bias: 2

This command is sent from VDBOX/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:23	**Media Command Opcode** Default Value: Ah Media Misc Format: OpCode
	22:21	**SubOpcodeA** Default Value: 0h Common Format: OpCode
	20:16	**SubOpcodeB** Default Value: 3h SFC_IEF_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 16h Excludes DWord (0,1) Format: =n Total Length - 2
1..23	735:0	**SFC IEF State Body** Format: SFC_IEF_STATE_BODY

SFC_LOCK

SFC_LOCK

Source: BSpec

Length Bias: 2

This command is used for VD/VE box to communicate with SFC before the start of any SFC workload. VD/VE uses this command to make sure that it has the ownership of SFC pipe before running workload with SFC since SFC is shared between VD/VE on a frame level.
For VD(MFX)-SFC workload, only decoder mode is allowed. Encoder mode cannot use SFC.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:23	**Media Command Opcode** Default Value: Ah Media Misc Format: OpCode
	22:21	**SubOpcodeA** Default Value: 0h Common Format: OpCode
	20:16	**SubOpcodeB** Default Value: 0h SFC Lock Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:0	**SFC Lock Body** Format: SFC_LOCK_BODY

SFC_STATE

SFC_STATE

Source: BSpec

Length Bias: 2

Description
This command is sent from VDBOX/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:23	**Media Command Opcode** Format: OpCode Value: Ah — Name: Media Misc **[Default]**
	22:21	**SubOpcodeA** Default Value: 0h Common Format: OpCode
	20:16	**SubOpcodeB** Default Value: 1h SFC_State Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Total Length - 2 Value: 1Eh — Name: Excludes DWord (0,1) **[Default]**
1	31:11	**Reserved**

Format:	MBZ

10:8 **VD/VE Input Ordering Mode**

Format:	U3

- VD mode: (SFC pipe mode set as "0")
- VE mode: (pipe mode set as "1 and 4")

For values for each mode, please refer to the table below:

Value	Name	Description	Exists If
0		16x16 block z-scan order - no shift	//VD Mode
1		16x16 block z-scan order - 4 pixels shift upward	//VD Mode
2		8x8 block jpeg z-scan order	//VD Mode
3		16x16 block jpeg z-scan order	//VD Mode
4		16x16 block VP8 row-scan order - no shift	//VD Mode
5-7		Reserved	//VD Mode
0		8x4 block column order, 64 pixel column	//VE Mode
1		4x4 block column order, 64 pixel column	//VE Mode
2		8x4 block column order, 128 pixel column	//VE Mode
3		4x4 block column order, 128 pixel column	//VE Mode
4-7		Reserved	//VE Mode

Programming Notes

This field shall be programmed according to video modes used in VDBOX. NOTE: SFC supports progressive input and output only (Interlaced/MBAFF is not supported).

Video Mode	Surface Format	SFC Input Chroma Sub-Sampling	VD/VE Input Ordering Mode
VC1 w/o LF and w/o OS Note: VC1 LF applies for either ILDB	420 (NV12)	1	0
VC1 w/ LF or w/ OS or w/ both Note: VC1 LF applies for either ILDB		INVALID with SFC	INVALID with SFC
AVC w/o LF	Monochrome	0	0
AVC w/o LF	420 (NV12)	1	0
AVC with LF	Monochrome	0	1
AVC/VP8 with LF	420 (NV12)	1	1
VP8 w/o LF	420 (NV12)	1	4
JPEG (YUV Interleaved)	Monochrome	0	2
JPEG (YUV Interleaved)	420	1	3
JPEG (YUV Interleaved)	422H_2Y	2	2

JPEG (YUV Interleaved)	422H_4Y	2	3
JPEG (YUV Interleaved)	444	4	2
JPEG (YUV Interleaved)	411	5	2

VEBOX MODE	VEBOX Single Pipe Enable Bit	SFC Input Surface Format	SFC Input Chroma Sub Sampling	VD/VE Input Ordering Mode
1. DN/HP with RGB input 2. Camera pipe (DM) enabled 3. CCM, FGC filters enabled	1	Monochrome	0	1
	1	420 (NV12)	1	1
	1	422H	2	1
	1	444	4	1
All other modes: (Legacy DN/DI features)	0	Monochrome	0	0
	0	420 (NV12)	1	0
	0	422H	2	0
	0	444	4	0

7:4 **SFC Input Chroma Sub-Sampling**

Value	Name	Description
0	4:0:0	SFC to insert UV channels
1	4:2:0	
2	4:2:2 Horizontal	VD: 2:1:1
3	Reserved	
4	4:4:4 Progressive/Interleaved	

Programming Notes

This field shall be programmed according to video modes used in VDBOX. NOTE: SFC supports progressive input and output only (Interlaced/MBAFF is not supported).

Video Mode	Surface Format	SFC Input Chroma Sub-Sampling	VD/VE Input Ordering Mode
VC1 w/o LF and w/o OS Note: VC1 LF applies for either ILDB	420 (NV12)	1	0
VC1 w/ LF or w/ OS or w/ both Note: VC1 LF applies for either ILDB		INVALID with SFC	INVALID with SFC
AVC w/o LF	Monochrome	0	0
AVC w/o LF	420 (NV12)	1	0
AVC with LF	Monochrome	0	1

AVC/VP8 with LF	420 (NV12)	1	1
VP8 w/o LF	420 (NV12)	1	4
JPEG (YUV Interleaved)	Monochrome	0	2
JPEG (YUV Interleaved)	420	1	3
JPEG (YUV Interleaved)	422H_2Y	2	2
JPEG (YUV Interleaved)	422H_4Y	2	3
JPEG (YUV Interleaved)	444	4	2

VEBOX MODE	Surface Format	SFC Input Chroma Sub Sampling	VD/VE Input Ordering Mode
Legacy DN/DI features	Monochrome	0	0
Legacy DN/DI features	420 (NV12)	1	0
Legacy DN/DI features	422H	2	0
Legacy DN/DI features	444	4	0
Capture/Camera pipe enabled(Demosaic)	Monochrome	0	1
Capture/Camera pipe enabled(Demosaic)	420 (NV12)	1	1
Capture/Camera pipe enabled(Demosaic)	422H	2	1
Capture/Camera pipe enabled(Demosaic)	444	4	1

3:0 **SFC Pipe Mode**

Value	Name	Description
0		VD-to-SFC AVS
1		VE-to-SFC AVS + IEF + Rotation
2-3		Reserved
4		VE-to-SFC Integral Image
5-15		Reserved

Programming Notes
Note: for SFC Pipe mode set to VE-to-SFC AVS mode. IECP pipeline mode MUST be enabled. However, each sub-IECP feature can be turned on/off independently.

2 31:28 **Reserved**

Format:	MBZ

27:16 **Input Frame Resolution Height**

Format:	U12-1

SFC_STATE

		Minus 1 in unit of pixel [11:0]. It is set to the value of the output resolution or number of pixels streaming into SFC from VD or VEBox. Since the max value support is 4k pixels, the max value allowed is 4K minus 1. • VDBOX frame height is multiple of 16 for Video source and JPEG formats other than 400, 444 and 422H_2Y. • VDBOX frame height is multiple of 8 for JPEG formats 400, 444 and 422H_2Y. • VEBOX frame height is multiple of 4.
		Min Resolution is 128 pixels. *Max Resolution* is up to 4K pixels. e.g. for 1920x1080 content, FrameHeightInMBsMinus1 is equal to1087 (1080 rounded up 16 pixel boundary, minus 1. i.e. effectively specified as 1088 instead).
		Restriction For Integral Image Mode, this field is Reserved and MBZ.
	15:12	**Reserved** Format: MBZ
	11:0	**Input Frame Resolution Width** Format: U12-1 Minus 1 in unit of pixel [11:0]. It is set to the value of the output resolution or number of pixels streaming into SFC from VD or VEBox. Since the max value support is 4k pixels, the max value allowed is 4K minus 1. • VDBOX frame width is multiple of 16 for Video source and JPEG formats other than 400, 444 and 422H_2Y. • VDBOX frame width is multiple of 8 for JPEG formats 400, 444 and 422H_2Y. • VEBOX frame width is multiple of 16. *Min Resolution* is 128 pixels. *Max Resolution* is up to 4K pixels. e.g. for 1920x1080 content, FrameHeightInMBsMinus1 is equal to1087 (1080 rounded up 16 pixel boundary, minus 1. i.e. effectively specified as 1088 instead). **Restriction** For Integral Image Mode, this field is Reserved and MBZ.
3	31:17	**Reserved** Format: MBZ
	16	**Reserved**

SFC_STATE

15:12 **Output Chroma Downsampling co-siting position Horizontal Direction**

Format:	U4

This field specifies the fractional position of the bilinear filter for chroma downsampling. In the X-axis.

Value	Name	Description
0000b	0/8 (Left full pixel)	0 (fraction_in_integer)
0001b	1/8	1 (fraction_in_integer)
0010b	1/4 (2/8)	2 (fraction_in_integer)
0011b	3/8	3 (fraction_in_integer)
0100b	1/2 (4/8)	4 (fraction_in_integer)
0101b	5/8	5 (fraction_in_integer)
0110b	3/4 (6/8)	6 (fraction_in_integer)
0111b	7/8	7 (fraction_in_integer)
1000b	8/8	

Programming Notes
For 444 format, horizontal chroma-siting should be programmed to zero.

11:8 **Output Chroma Downsampling co-siting position Vertical Direction**

Format:	U4

This field specifies the fractional position of the bilinear filter for chroma downsampling. In the Y-axis.

Value	Name	Description
0000b	0/8 (Left full pixel)	0 (fraction_in_integer)
0001b	1/8	1 (fraction_in_integer)
0010b	1/4 (2/8)	2 (fraction_in_integer)
0011b	3/8	3 (fraction_in_integer)
0100b	1/2 (4/8)	4 (fraction_in_integer)
0101b	5/8	5 (fraction_in_integer)
0110b	3/4 (6/8)	6 (fraction_in_integer)
0111b	7/8	7 (fraction_in_integer)
1000b	8/8	

Programming Notes
For 444 and 422 format, vertical chroma co-siting value should be programmed to zero.

7:6 **Reserved**

Format:	MBZ

5 **RGBA_Channel_Swap Enable**

Default Value:	0
Format:	Enable

This bit should only be used with RGB output formats and CSC conversion is turned on. When this bit is set, the R and B channels are swapped into the output RGB channels as shown in the following table:

Name	Bits	MSB Color Order	Swapped
RGBA8	8:8:8:8	A:B:G:R	A:R:G:B
RGBA10	2:10:10:10	A:R:G:B	A:B:G:R
RGB 5:6:5	5:6:5	R:G:B	B:G:R

4 **Reserved**

Format:	MBZ

3:0 **Output Surface Format type**

SFC output surface format type.
Reserved

Value	Name	Description	Exists If
0		AYUV 4:4:4 (8:8:8:8 MSB-A:Y:U:V)	//Tile-Y/ Tile-X/Linear
1		RGBA8 4:4:4:4 (8:8:8:8 MSB-A:B:G:R)	//Tile-Y/ Tile-X/Linear
2		RGBA10 10:10:10:2 (2:10:10:10 MSB-A:R:G:B)	//Tile-Y/ Tile-X/Linear
3		RGB 5:6:5 (5:6:5 MSB-R:G:B)	//Tile-Y/ Tile-X/Linear
4		Planar NV12 4:2:0 8-bit	//Tile-Y
5		Packed YUYV 4:2:2 8-bit	//Tile-Y/ Tile-X/Linear
6		Packed UYVY 4:2:2 8-bit	//Tile-Y/ Tile-X/Linear
7		Packed integral Image 32-bit	//Linear
8		Packed integral Image 64-bit	//Linear
9		P016 format	//Tile-Y

Restriction
For Integral Image Mode, output surface format type must be set to 32/64-bit Integral Image Plane. Driver/SW must ensure the max accumulated integral image value does not exceed the programmable output precision. HW will simply generate wrong value once it overflow in wrap around case.

SFC_STATE

4 | 31:22 | Reserved

Format:	MBZ

21:20 | BitDepth

Description
This field is valid only for output formats P016. This field is used to specify how many of the LSB bits have valid data.

Value	Name	Description
0	10BitFormat	Higher 10 bits are valid and lower 6 bits are 0
1	16BitFormat	Higher 12 bits are valid and lower 4 bits are 0

19 | CSC Enable

Description
This field specifies the scaled pixels are converted from YUV to RGB. The output format type must be programmed in one of the RGBAx type.

Restriction
For Integral Image Mode, this field is Reserved and MBZ.

18 | Color Fill Enable

Programming Notes
This field could be enabled only if the scaled resolution is smaller than the output/display resolution. If enabled, HW will fill the gap with programmable pixel values. Else, nothing will be filled in the gap region.
Usage: Color fill must be enabled for the first time/pass when a new surface is allocated/ used. Optional for subsequence frames since the gap region is filled with default pixels by prior passes.

17:16 | Rotation Mode

Format:	U2

Value	Name
00b	0 (degrees)
01b	90 Clockwise
10b	180 Clockwise
11b	270 Clockwise

Programming Notes
SFC rotation (90, 180 and 270) should be set only on VEBox input mode and SFC output set to TileY. Restriction: • For Integral Image Mode, this field is Reserved and MBZ. • For VDBox Mode, this field is Reserved and MBZ. • For linear or TileX SFC output, this field is Reserved and MBZ.

15:13 **Reserved**

Format:	MBZ

12 **Chroma Upsampling Enable**

This field enables the high-quality UV channel upsampler prior to IEF filter process. This field should be disabled when the source pixels and output pixels are kept with the same chroma sub-sample type and IEF is disabled.

Restriction
For Integral Image Mode, this field is Reserved and MBZ.

11:10 **Reserved**

Format:	MBZ

9 **Bypass X Adaptive Filtering**

Value	Name	Description
0	Enable X Adaptive Filtering	
1	Disable X Adaptive Filtering	The X direction will use **Default Sharpness Level** to blend between the smooth and sharp filters rather than the calculated value.

8 **Bypass Y Adaptive Filtering**

Value	Name	Description
0	Enable Y Adaptive Filtering	
1	Disable Y Adaptive Filtering	The Y direction will use **Default Sharpness Level** to blend between the smooth and sharp filters rather than the calculated value.

7 **AVS Scaling Enable**

Value	Name	Description
1	Enable	
0	Disable	The scaling factor is ignored and a scaling ratio of 1:1 is assumed.

6 **Adaptive Filter for all Channels**

SFC_STATE

Value	Name	Description
1	Enable Adaptive Filter on UV/RB Channels	8-tap Adaptive Filter Mode is on
0	Disable Adaptive Filter on UV/RB Channels	

5:4 **AVS Filter Mode**

Value	Name
0	5x5 Poly-phase filter + Bilinear (adaptive)
1	8x8 poly-phase filter + Bilinear (adaptive)
2	Bilinear filter only
3	Reserved

Programming Notes
In VD-to-SFC mode, value of 1 is not allowed.

3 **Reserved**

Format:	MBZ

2 **IEF4Smooth_Enable**

Value	Name	Description
0	**[Default]**	IEF is operating as a content adaptive detail filter based on 5x5 region.
1		IEF is operating as a content adaptive smooth filter based on 3x3 region

Restriction
For Integral Image Mode, this field is Reserved and MBZ.

1 **Skin Tone Tuned IEF_Enable**

Exists If:	//IEF Enable = 1

Restriction
For Integral Image Mode, this field is Reserved and MBZ.

0 **IEF Enable**

Value	Name	Description
1	Enable	IEF Filter is Enabled
0	Disable	IEF Filter is Disabled

Restriction
For Integral Image Mode and VD Mode, this field is Reserved and MBZ.

5 31:28 **Reserved**

		Format: MBZ
	27:16	**Source Region Height** Format: U12-1 **Source/Crop Region Height Minus 1 of the Input Frame in Unit of Pixel [11:0].** This field specifies the source/crop region of the input frame used for scaling of the graphic view. It defines the out-of-frame boundary used prior to AVS/IEF interpolation operation. The max value should be programmed to be equal or small than the input FrameHeightinMBminus 1 field. e.g. for 1920x1080 content, FrameHeightInMBsMinus1 is equal to1087 (1088 lines); however, the crop region height should be set to 1079(1080 lines). The last 8 lines are assumed to be not usable and should not be used as source pixels for Scaling or IEF operations. Otherwise, the bad pixels will breach and cause artifacts into the scaled output frame. **Restriction** For Integral Image Mode, this field is Reserved and MBZ. For AVS mode, the restriction is tied to chroma input format type: 420 - multiple of 2. 422/444/400 - no restrictions, except for AVS bypass case (ie. 1:1 scaling) where restriction is tied to chroma output format. **Min Resolution** is 128 pixels. **Max Resolution** is 4K pixels.
	15:12	**Reserved** Format: MBZ
	11:0	**Source Region Width** Format: U12-1 **Source/Crop Region Width Minus 1 of the Input Frame in Unit of Pixel [11:0].** This field specifies the source/crop region of the input frame used for scaling of the graphic view. It defines the out-of-frame boundary used prior to AVS/IEF interpolation operation. The max value should be programmed to be equal or small than the input FrameWidthinMBminus 1 field. e.g. for 1920x1080 content, FrameWidthInMBsMinus1 is equal to1919 (1920 pixel wide); however, the crop region width should be set to less than 1909(1910 pixel wide). The last 10 pixels of the frame are assumed to be not usable and should not be used as source pixels for Scaling or IEF operations. Otherwise, the bad pixels will breach and cause artifacts into the scaled output frame. **Restriction** For Integral Image Mode, this field is Reserved and MBZ. For AVS mode, the restriction is tied to chroma input format type: 420 - multiple of 2. 422 - multiple of 2. 444/400 - no restrictions, except for AVS bypass case (ie. 1:1 scaling) where restriction is tied to chroma output format.**Min Resolution** is 128 pixels. **Max Resolution** is 4K pixels.
6	31:28	**Reserved**

SFC_STATE

		Format: MBZ
	27:16	**Source Region Vertical Offset** Format: U12 **Vertical Offset Of The SRC Region Relative To The Starting Position Of The Input Frame In Unit Of Pixel [11:0]** This field specifies the vertical offset of the starting position of the scaled region relatives to the starting position (pixel 0,0) of the output frame. It defines the out-of-frame boundary used prior to AVS/IEF interpolation operation. This value should be set to zero if the starting corner of the crop region is same as the input frame region. The sum of this value and the src/crop region size heightminus1 must be programmed to be equal or small than the input FrameHeightinMBminus 1 field. **Restriction** For Integral Image Mode, this field is Reserved and MBZ. For AVS mode, the restriction is tied to chroma input format type: 420 - multiple of 2. 422/444/400 - no restrictions.
	15:12	**Reserved** Format: MBZ
	11:0	**Source Region Horizontal Offset** Format: U12 **Horizontal Offset Of The SRC Region Relative To The Starting Position Of The Input Frame In Unit Of Pixel [11:0]** This field specifies the horizontal offset of the starting position of the scaled region relatives to the starting position (pixel 0,0) of the output frame. It defines the out-of-frame boundary used prior to AVS/IEF interpolation operation. This value should be set to zero if the starting corner of the crop region is same as the input frame region. The sum of this value and the src/crop region size widthminus1 must be programmed to be equal or small than the input FrameWidthinMBminus 1 field. **Restriction** For Integral Image Mode, this field is Reserved and MBZ. For AVS mode, the restriction is tied to chroma input format type: 420 - multiple of 2. 422 - multiple of 2. 444/400 - no restrictions.
7	31:28	**Reserved** Format: MBZ
	27:16	**Output Frame Height**

SFC_STATE

		Format: U12-1 It is set to the value of the final output resolution of the graphic view. Since the max value support is 4k pixels, the max value allowed is 4K minus 1. **Restriction** For Integral Image Mode, this field is Reserved and MBZ. For AVS mode, the restriction is tied to chroma output format type: 420 - multiple of 2. 422/444/400 - no restrictions. **Min Resolution** is 128 pixels. **Max Resolution** is 4K pixels.
	15:12	**Reserved** Format: MBZ
	11:0	**Output Frame Width** Format: U12-1 It is set to the value of the final output resolution of the graphic view. Since the max value support is 4k pixels, the max value allowed is 4K minus 1. **Restriction** For Integral Image Mode, this field is Reserved and MBZ. For AVS mode, the restriction is tied to chroma output format type: 420 - multiple of 2. 422 - multiple of 2. 444/400 - no restrictions. **Min Resolution** is 128 pixels. **Max Resolution** is 4K pixels.
8	31:28	**Reserved** Format: MBZ
	27:16	**Scaled Region Size Height** Format: U12-1 It is set to the height of the scaled region over the output frame of the graphic view. **Programming Notes** The Max Value =< [The Output Frame Height Minus1]. **Restriction** For AVS mode, if rotation_mode = 0/180, the restriction is tied to chroma output format type: 420 - multiple of 2. 422/444/400 - no restrictions. For AVS mode, if rotation_mode = 90/270, the restriction is tied to chroma output format type: 420/422 - multiple of 2. 444/400 - no restrictions. **Min Resolution** is 128 pixels. **Max Resolution** is 4K pixels.
	15:12	**Reserved**

		Format: MBZ
	11:0	**Scaled Region Size Width** Format: U12-1 It is set to the Width of the scaled region over the output frame of the graphic view. **Programming Notes** The Max Value =< [The Output Frame Width Minus1]. **Restriction** For AVS mode, the restriction is tied to chroma output format type: 420 - multiple of 2. 422 - multiple of 2. 444/400 - no restrictions. **Min Resolution** is 128 pixels. **Max Resolution** is 4K pixels.
9	31:29	**Reserved** Format: MBZ
	28:16	**Scaled Region Vertical Offset** Format: S12 **Vertical Offset (in pixels) Of The Scaled Region Relatives to The Starting Position Of The Output Frame In Unit Of Pixel [11:0]** This field specifies the vertical offset of the starting position of the scaled region relatives to the starting position (pixel 0,0) of the output frame. The gap between the scaled and output frame shall be filled by hardware with a set of programmed YUV/RGB values (Grey Bar). This value should be set to zero if the starting corner of the scaled region is same as the output frame region. The sum of this value and the scaled region size Heightminus1 must be programmed to be equal or small than the output FrameHeightinMBminus 1 field plus 16. **Programming Notes** This field must be set to zero if SFC Output surface format type is P010/P016. **Restriction** For Integral Image Mode, this field is Reserved and MBZ. For AVS mode, the restriction is tied to chroma output format type: 420 - multiple of 2. 422/444/400 - no restrictions. This field must be set to zero if SFC Output surface format type is NV12. When mirror mode is set to 1, this field should be programmed to value greater than or equal to zero. The sum of this value and the Scaled Region Size Height must be smaller than or equal the Output Frame Height. The Scaling Factor Height needs to be programmed according to the scaled region within the frame.
	15:13	**Reserved**

		Format: MBZ
	12:0	**Scaled Region Horizontal Offset** Format: S12 **Horizontal Offset (in pixels) Of The Scaled Region Relatives to The Starting Position Of The Output Frame In Unit Of Pixel [11:0]** This field specifies the horizontal offset of the starting position of the scaled region relatives to the starting position (pixel 0,0) of the output frame. The gap between the scaled and output frame shall be filled by hardware with a set of programmed YUV/RGB values (Grey Bar). This value should be set to zero if the starting corner of the scaled region is same as the output frame region. The sum of this value and the scaled region size Widthminus1 must be programmed to be equal or small than the output FrameWidthinMBminus 1 field plus 16. **Programming Notes** This field must be set to zero if SFC Output surface format type is P010/P016. **Restriction** For Integral Image Mode, this field is Reserved and MBZ. For AVS mode, the restriction is tied to chroma output format type: 420 - multiple of 2. 422 - multiple of 2. 444/400 - no restrictions. This field must be set to zero if SFC Output surface format type is NV12. When mirror mode is set to 1, this field should be programmed to value greater than or equal to zero. The sum of this value and the Scaled Region Size Width must be smaller than or equal the Output Frame Width. The Scaling Factor Width needs to be programmed according to the scaled region within the frame.
10	31:26	**Reserved** Format: MBZ
	25:16	**Gray Bar Pixel - Y/R** Format: 10-bit UNORM Type **Range:**[0.0, +1.0] This is the default value used to fill in the area between the scaled region and the output frame size (aka Gray Bar) in Y or R channel on the AYUV or RGBA domain respectively. **Restriction** For Integral Image Mode, this field is Reserved and MBZ.
	15:10	**Reserved** Format: MBZ
	9:0	**Gray Bar Pixel - U/G**

		Format: 10-bit UNORM Type **Range:**[0.0, +1.0] This is the default value used to fill in the area between the scaled region and the output frame size (aka Gray Bar) in U or G channel on the AYUV or RGBA domain respectively. **Restriction** For Integral Image Mode, this field is Reserved and MBZ.
11	31:26	**Reserved** Format: MBZ
	25:16	**Gray Bar Pixel - V/B** Format: 10-bit UNORM Type **Range:**[0.0, +1.0] This is the default value used to fill in the area between the scaled region and the output frame size (aka Gray Bar) in V or B channel on the AYUV or RGBA domain respectively. **Restriction** For Integral Image Mode, this field is Reserved and MBZ.
	15:10	**Reserved** Format: MBZ
	9:0	**Gray Bar Pixel - A** Format: 10-bit UNORM Type **Range:**[0.0, +1.0] This is the default value used to fill in the area between the scaled region and the output frame size (aka Gray Bar) in A channel on the AYUV or RGBA domain respectively. **Restriction** For Integral Image Mode, this field is Reserved and MBZ.
12	31:26	**Reserved** Format: MBZ
	25:16	**UV Default value for V channel (For Mono Input Support)** Exists If: //Input NOT originated by VEBOX. Format: 10-bit UNORM Type **Range:**[0.0, +1.0] This field specifies the UV default value fill in to the UV output channels when input from

SFC_STATE

		VDBOX is set to Monochrome. **Restriction** Not used when input is originated by VEBOX (Including Integral Image Mode).
	15:10	**Reserved** Format: MBZ
	9:0	**UV Default value for U channel (For Mono Input Support)** Exists If: //Input NOT originated by VEBOX. Format: 10-bit UNORM Type **Range:**[0.0, +1.0] This field specifies the UV default value fill in to the UV output channels when input from VDBOX is set to Monochrome. **Restriction** Not used when input is originated by VEBOX (Including Integral Image Mode).
13	31:10	**Reserved** Format: MBZ
	9:0	**Alpha Default Value** Format: 10-bit UNORM Type **Range:**[0.0, +1.0] This field specifies the Alpha default value fill into the alpha output channel when output format type is set to RGBA8/10. **Restriction** For Integral Image Mode, this field is Reserved and MBZ.
14	31:21	**Reserved** Format: MBZ
	20:0	**Scaling Factor Height** Format: U4.17 This field specifies the scaling ratio of the vertical sizes between the crop/source region and the scaled region. The destination pixel coordinate, y-axis, is multiplied with this scaling factor to mapping back to the source input pixel coordinate. The field specifies the ratio of crop height resolution/ scaled height resolution. This implies $1/sf_v$ in the equation.

		Value / **Name** / **Description** 0 / / Reserved **Restriction** Integral Image Mode supports downsampling only.
15	31:21	**Reserved** Format: MBZ
	20:0	**Scaling Factor Width** Format: U4.17 This field specifies the scaling ratio of the horizontal sizes between the crop/source region and the scaled region. The destination pixel coordinate, x-axis, is multiplied with this scaling factor to mapping back to the source input pixel coordinate. The field specifies the ratio of crop width resolution/ scaled width resolution. This implies $1/sf_u$ in the equations above. **Value** / **Name** / **Description** 0 / / Reserved **Programming Notes** When DI-AVS is enabled in VEBOX+SFC mode,2X scaling is done in VEBOX which restricts the SFCScale Factor Width to do only 4X. In such case, Scaling Factor Width cannot be less than1/4. **Restriction** Integral Image Mode supports downsampling only.
16	31:0	**Reserved** Format: MBZ
17	31:12	**Output Frame Surface Base Address** Specifies the 4K byte aligned frame buffer address for outputting the scaled up/down image. Data is stored in Tile-Y format. For Integral Image mode, the accumulated integral image values will be packed linear in this surface. **Programming Notes** This field is ignored if I-frame only mode is set to 0 (Disable).
	11:0	**Reserved** Format: MBZ
18	31:16	**Reserved**

		Format: MBZ
	15:0	**Output Frame Surface Base Address High** This field is for the upper range [47:32] of Output Frame Surface Base Address. For Integral Image mode, the accumulated integral image values will be packed linear in this surface.
19	31:15	**Reserved** Format: MBZ
	14:13	**Output Surface Tiled Mode** Format: U2 **For Media Surfaces:** This field specifies the tiled resource mode.

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

	12	**Output Frame Surface Base Address - Row Store Scratch Buffer Cache Select** Format: Enable

Value	Name	Description
0	Disable **[Default]**	This field must be programmed to 0

Programming Notes
This must be set to 0

	11	**Reserved** Format: MBZ
	10	**Output Frame Surface Base Address - Memory Compression Mode** Format: U1

Description
Distinguishes vertical from horizontal compression. Please refer to vol1a**Memory Data Formats chapter – section** media Memory Compression for more details.

Value	Name	Programming Notes
0	Vertical Compression	[] Recommendation to use vertical compression

				VE-SFC: 0.5x scaling on horizontal direction or below; VD-SFC: 2x scaling on horizontal direction or below
		1	Horizontal Compression	[] Recommendation to use vertical compression VE-SFC: 0.5x scaling on horizontal direction or above VD-SFC: 2x scaling on horizontal direction or above

	9	**Output Frame Surface Base Address - Memory Compression Enable** Format: Enable Memory compression will be attempted for this surface.
	8:7	**Output Frame Surface Base Address - Arbitration Priority Control** Format: **HEVC_ARBITRATION_PRIORITY**
	6:1	**Output Frame Surface Base Address - Index to Memory Object Control State (MOCS) Tables** Format: U6 The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.
	0	**Reserved**
20	31:12	**AVS Line Buffer Surface Base Address** Specifies the 4K byte aligned frame buffer address for scratch space used for row/column store. This surface is used only if the internal buffer inside the SFC HW is not large enough to contain all row/column memory accesses. The AVS line buffer needs to be a valid address even for 1:1 scaling if SFC is used.
	11:0	**Reserved** Format: MBZ
21	31:16	**Reserved** Format: MBZ
	15:0	**AVS Line Buffer Surface Base Address High** This field is for the upper range [47:32] of AVS Line Buffer Surface Base Address. AVS Line buffer address needs to be valid even for 1:1 scaling if SFC is used.
22	31:15	**Reserved** Format: MBZ
	14:13	**AVS Line Buffer Tiled Mode** Format: U2 **For Media Surfaces:** This field specifies the tiled resource mode.

SFC_STATE

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

12 **AVS Line Buffer Base Address - Row Store Scratch Buffer Cache Select**

Format:	U1

This field controls if the Row Store is going to store inside Media Cache (rowstore cache) or to LLC.

Value	Name	Description
0	LLC **[Default]**	Buffer going to LLC

Programming Notes
This suface does not support to put in Row Store Scratch Buffer. Must be set to 0

11 **Reserved**

Format:	MBZ

10 **AVS Line Buffer Base Address - Memory Compression Mode**

Default Value:	0 Horizontal Compression Mode
Format:	U1

Distinguishes vertical from horizontal compression. Please refer to vol1a **Memory Data Formats chapter - section** media Memory Compression for more details.

Programming Notes
Memory compression is not supported. This bit is not used. Default to 0

9 **AVS Line Buffer Base Address - Memory Compression Enable**

Default Value:	0 Disable
Format:	Enable

This bit control memory compression for this surface

Programming Notes
This bit must be set to 0 (Memory compression is not supported in this surface)

8:7 **AVS Line Buffer Base Address - Arbitration Priority Control**

Format:	HEVC_ARBITRATION_PRIORITY

6:1 **AVS Line Buffer Base Address - Index to Memory Object Control State (MOCS) Tables**

Format:	U6

The index to define the L3 and system cache memory properties. The details of the controls are

		further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.
	0	**Reserved**
23	31:12	**IEF Line Buffer Surface Base Address** Specifies the 4K byte aligned frame buffer address for the scratch space used for row/column store. This surface is used only if the internal buffer inside SFC HW is not large enough to contain all row/column memory accesses. **Restriction** For Integral Image Mode, this field is Reserved and MBZ.
	11:0	**Reserved** Format: MBZ
24	31:16	**Reserved** Format: MBZ
	15:0	**IEF Line Buffer Surface Base Address High** This field is for the upper range [47:32] of IEF Line Buffer Surface Base Address. **Restriction** For Integral Image Mode, this field is Reserved and MBZ.
25	31:15	**Reserved** Format: MBZ
	14:13	**IEF Line Buffer Tiled Mode** Format: U2 **For Media Surfaces:** This field specifies the tiled resource mode. **Value** / **Name** / **Description** 0h / TRMODE_NONE / No tiled resource 1h / TRMODE_TILEYF / 4KB tiled resources 2h / TRMODE_TILEYS / 64KB tiled resources 3h / Reserved /
	12	**IEF Line Buffer Base Address - Row Store Scratch Buffer Cache Select** Format: U1 This field controls if the Row Store is going to store inside Media Cache (rowstore cache) or to LLC. **Value** / **Name** / **Description** 0 / LLC **[Default]** / Buffer going to LLC **Programming Notes**

		This surface does not support Rowstore Scratch Buffer Cache. Must be programmed to 0
	11	**Reserved** Format: MBZ
	10	**IEF Line Buffer Base Address - Memory Compression Mode** Default Value: 0 Format: U1 Distinguishes vertical from horizontal compression. Please refer to vol1a **Memory Data Formats chapter - section** media Memory Compression for more details. **Programming Notes** Must be zero; memory compression is not supported for this surface. Default to0
	9	**IEF Line Buffer Base Address - Memory Compression Enable** Default Value: 0 Disable Format: Enable **Programming Notes** Memory compression is not supported for this surface Must be 0.
	8:7	**IEF Line Buffer Base Address - Arbitration Priority Control** Format: HEVC_ARBITRATION_PRIORITY
	6:1	**IEF Line Buffer Base Address - Index to Memory Object Control State (MOCS) Tables** Format: U6 The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.
	0	**Reserved**
26	31:0	**Reserved** Format: MBZ
27	31:0	**Reserved** Format: MBZ
28	31:0	**Reserved** Format: MBZ
29	31:28	**Output Surface Format**

SFC_STATE

	27	**Output Surface Interleave Chroma Enable**
	26:20	**Reserved** Format: MBZ
	19:3	**Output Surface Pitch** Format: U17-1 Pitch in (Bytes - 1) This field specifies the surface pitch.

Value	Name	Description
[0,2047]	SURFTYPE_BUFFER Surfaces	[1B, 2048B]
[0, 524287]	Other Linear Surfaces	[64B, 512KB] = [1 CL, 8K CLs]
[511, 524287]	X-tiled Surface	[512B, 256KB] = [1tile, 512 tiles]
[127, 524287]	Y-tiled surfaces	[128B,256KB] = [1 tile, 2048 tiles]

Programming Notes

- For tiled surfaces, the pitch must be a multiple of the tile width
- For Linear surfaces, the pitch must be a multiple of CL (64B) width
- If **Half Pitch for Chroma** is set, this field must be a multiple of two tile widths for tiled surfaces, or a multiple of 2 bytes for linear surfaces.

If Media Memory Compression is enabled, the following max pitch size restriction must be honored. For larger resolution, Media Memory compression Must be disabled.

Tiling Mode	Pixel Format	Max Frame Width (bytes)	Max Frame Width (pixels)	Max Pitch (bytes)
Legacy 4K	8bpp	16k	16k	16k + 127
	16bpp	16k	8k	16k + 127
	32bpp	16k	4k	16k + 127
	64bpp	16k	2k	16k + 127
	128bpp	16k	1k	16k + 127
TileYF	8bpp	8k	8k	8k + 63
	16bpp	16k	8k	16k + 127
	32bpp	16k	4k	16k + 127
	64bpp	16k	2k	16k + 255
	128bpp	16k	1k	16k + 255
TileYS	8bpp	16k	16k	16k + 255
	16bpp	16k	8k	16k + 511
	32bpp	16k	4k	16k + 511

		64bpp	16k	2k	16k + 1023
		128bpp	16k	1k	16k + 1023

2 **Output Surface Half Pitch For Chroma**

Exists If:	//PLANAR Surface Formats Only
Format:	Enable

This field indicates that the chroma plane(s) will use a pitch equal to half the value specified in the Surface Pitch field.

1 **Output Surface Tiled**

Format:	Boolean

This field specifies whether the surface is tiled.

Value	Name	Description
1	True	Tiled
0	FALSE	Linear

Programming Notes

- Linear surfaces can be mapped to Main Memory (uncached) or System Memory (cacheable, snooped). Tiled surfaces can only be mapped to Main Memory.
- The corresponding cache(s) must be invalidated before a previously accessed surface is accessed again with an altered state of this bit.
- The tiled surfaces of current picture and reference picture should be declared as the identical type in VDI mode with the identical Height, Width and Format.

0 **Output Surface Tile Walk**

Format:	SFC_Tile_Walk

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.

Value	Name
0	TILEWALK_XMAJOR
1	TILEWALK_YMAJOR

Programming Notes

- The corresponding cache(s) must be invalidated before a previously accessed surface is accessed again with an altered state of this bit.

This field is ignored when the surface is linear.

30 31:30 **Reserved**

		Format: MBZ
	29:16	**Output Surface X Offset For U**
		Exists If: //PLANAR Surface Formats Only Format: U14 Pixel Offset This field specifies the horizontal offset in pixels from the **Surface Base Address** to the start (origin) of the U(Cb) plane or the interleaved UV plane if **Interleave Chroma** is enabled. **Programming Notes** For PLANAR_420 and PLANAR_422 surface formats, this field must be zero.
	15:14	**Reserved**
		Format: MBZ
	13:0	**Output Surface Y Offset For U**
		Exists If: //PLANAR Surface Formats Only Format: U14 Pixel Row Offset This field specifies the vertical offset in rows from the **Surface Base Address** to the start (origin) of the U(Cb) plane or the interleaved UV plane if **Interleave Chroma** is enabled. **Programming Notes** For PLANAR_420 and PLANAR_422 surface formats, this field must be multiple of 16 pixels - i.e. multiple MBs.
31	31:30	**Reserved**
		Format: MBZ
	29:16	**Output Surface X Offset For V**
		Exists If: //PLANAR Surface Formats with Interleaved Chroma Disable Format: U14 Pixel Offset This field specifies the horizontal offset in pixels from the **Surface Base Address** to the start (origin) of the V(Cr) plane. **Programming Notes** For PLANAR_420 and PLANAR_422 surface formats, this field must indicate an even number of pixels.
	15:14	**Reserved**
		Format: MBZ

<table>
<tr><th colspan="3">SFC_STATE</th></tr>
<tr><td></td><td>13:0</td><td>Output Surface Y Offset For V
<table>
<tr><td></td><td></td></tr>
<tr><td>Exists If:</td><td>//PLANAR Surface Formats with Interleaved Chroma Disable</td></tr>
<tr><td>Format:</td><td>U14 Pixel Offset</td></tr>
</table>
This field specifies the vertical offset in rows from the Surface Base Address to the start (origin) of the V(Cr) plane.
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>For PLANAR_420 and PLANAR_422 surface formats, this field must indicate an even number of pixels.</td></tr>
</table></td></tr>
<tr><td>32</td><td>31:1</td><td>Reserved</td></tr>
<tr><td></td><td>0</td><td>Reserved
<table>
<tr><td></td><td></td></tr>
</table></td></tr>
<tr><td>33</td><td>31:0</td><td>Reserved</td></tr>
</table>

Shift Left

shl - Shift Left

Source: Eulsa

Length Bias: 4

Description

Perform component-wise logical left shift of the bits in src0 by the shift count indicated in src1, storing the results in dst, inserting zero bits in the number of LSBs indicated by the shift count. Hardware detects overflow properly and uses it to perform any saturation operation on the result, as long as the shifted result is within 33 bits. Otherwise, the result is undefined. Note: For word and DWord operands, the accumulators have 33 bits.

In QWord mode, the shift count is taken from the low six bits of src1 regardless of the src1 type and treated as an unsigned integer in the range 0 to 63. Otherwise the shift count is taken from the low five bits of src1 regardless of the src1 type and treated as an unsigned integer in the range 0 to 31. The operation uses QWord mode if src0 or dst has the Q or UQ type but not if src1 is the only operand with the Q or UQ type.

Format:

```
[(pred)] shl[.cmod] (exec_size) dst src0 src1
```

Restriction

Accumulator cannot be destination, implicit or explicit.

Syntax

```
[(pred)] shl[.cmod] (exec_size) reg reg reg
[(pred)] shl[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
          for ( n = 0; n < exec_size; n++ ) {
             if ( WrEn.chan[n] ) {
                shiftCnt = src0 or dst has Q or UQ type ? src1.chan[n] & 0x3F :
src1.chan[n] & 0x1F
                dst.chan[n] = src0.chan[n] « shiftCnt;
             }
          }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

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

DWord	Bit	Description
0..3	127:64	**RegSource**

shl - Shift Left

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

Shift Right

shr - Shift Right

Source: Eulsa

Length Bias: 4

Description

Perform component-wise logical right shift with zero insertion of the bits in src0 by the shift count indicated in src1, storing the results in dst. Insert zero bits in the number of MSBs indicated by the shift count. src0 and dst can have different types and can be signed or unsigned. Note: For word and DWord operands, the accumulators have 33 bits. Note: For unsigned src0 types, shr and asr produce the same result.

In QWord mode, the shift count is taken from the low six bits of src1 regardless of the src1 type and treated as an unsigned integer in the range 0 to 63. Otherwise the shift count is taken from the low five bits of src1 regardless of the src1 type and treated as an unsigned integer in the range 0 to 31. The operation uses QWord mode if src0 or dst has the Q or UQ type but not if src1 is the only operand with the Q or UQ type.

Format:

```
[(pred)] shr[.cmod] (exec_size) dst src0 src1
```

Syntax

```
[(pred)] shr[.cmod] (exec_size) reg reg reg
[(pred)] shr[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
          for ( n = 0; n < exec_size; n++ ) {
             if ( WrEn.chan[n] ) {
                shiftCnt = src0 or dst has Q or UQ type ? src1.chan[n] & 0x3F :
src1.chan[n] & 0x1F
                dst.chan[n] = src0.chan[n] » shiftCnt;
             }
          }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types
UB, UW, UD	UB, UW, UD
UW, UD	UW, UD

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM') Format: **EU_INSTRUCTION_SOURCES_REG_REG**

shr - Shift Right

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

Signal Event MSD

MSD_SIGNAL_EVENT - Signal Event MSD

Source: EuSubFunctionGateway

Length Bias: 1

Sends an event to all threads that are monitoring that Event ID.

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Default Value: 0b Format: Enable **Restriction** Must be zero.
	18:3	**Reserved** Format: MBZ
	2:0	**Signal Event Subfunction** Default Value: 001b Format: OpCode

SIMD8 Render Target Read MSD

MSD_RTR_SIMD8 - SIMD8 Render Target Read MSD

Source: EuSubFunctionRenderDataPort
Length Bias: 1
Family: Other
Group: Render Target R/W

DWord	Bit	Description
0	31	**Reserved** Format: MBZ Ignored
	30	**Message Precision Subtype** Default Value: 0h Format: Opcode Full precision data message **Programming Notes** This field must be programmed to 0
	29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.

MSD_RTR_SIMD8 - SIMD8 Render Target Read MSD

	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the 2-register header.
	18	**Reserved** Format: MBZ Ignored
	17:14	**Message Type** Default Value: 0Dh Format: Opcode Render Target Read message
	13	**Per-Sample PS Enable** Format: Enable If set, PS reads color phases on per sample basis for each slot. **Programming Notes** This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE. When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0. When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples). When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.
	12	**Reserved** Format: MBZ Ignored
	11	**Slot Group Select**

MSD_RTR_SIMD8 - SIMD8 Render Target Read MSD

		Format: MDC_RT_SGS This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
	10:9	**Reserved** Format: MBZ Ignored
	8	**Render Target Message Subtype** Default Value: 1h Format: Opcode SIMD8 single source message. Use slots [7:0] for the pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [23:16] are referenced instead of [7:0].
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

SIMD8 Render Target Write MSD

MSD_RTW_SIMD8 - SIMD8 Render Target Write MSD

Source: EuSubFunctionRenderDataPort
Length Bias: 1
Family: Other
Group: Render Target R/W

DWord	Bit	Description
0	31	**Reserved** Format: MBZ Ignored
	30	**Message Precision Subtype** Default Value: 0h Format: Opcode Full precision data message
	29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present**

	Format:	MDC_MHP

If set, indicates that the message includes the 2-register header.

18 Per-Coarse Pixel PS outputs enable

Format:	Enable

This bit indicates the render target write is a coarse pixel write.

Programming Notes

This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.

17:14 Message Type

Default Value:	0Ch
Format:	Opcode

Render Target Write message

13 Per-Sample PS Enable

Format:	Enable

If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot.

Programming Notes

This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE.
When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0.
When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples).
When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.

12 Last Render Target Select

Format:	Enable

This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.

11 **Slot Group Select**

Format:	MDC_RT_SGS

This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.

10:8 **Render Target Message Subtype**

Default Value:	4h
Format:	Opcode

SIMD8 single source message. Use slots [7:0] for pixel enables, X/Y addresses, and oMask.

Programming Notes

The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [23:16] are referenced instead of [7:0].

7:0 **Binding Table Index**

Format:	MDC_BTS

Specifies the Binding Table Index for the message

SIMD16 Render Target Read MSD

MSD_RTR_SIMD16 - SIMD16 Render Target Read MSD

Source: EuSubFunctionRenderDataPort
Length Bias: 1
Family: Other
Group: Render Target R/W

DWord	Bit	Description
0	31	**Reserved** Format: MBZ Ignored
	30	**Message Precision Subtype** Default Value: 0h Format: Opcode Full precision data message **Programming Notes** This field must be programmed to 0
	29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.

MSD_RTR_SIMD16 - SIMD16 Render Target Read MSD

19	**Header Present**	
	Format:	MDC_MHP
	If set, indicates that the message includes the 2-register header.	
18	**Reserved**	
	Format:	MBZ
	Ignored	
17:14	**Message Type**	
	Default Value:	0Dh
	Format:	Opcode
	Render Target Read message	
13	**Per-Sample PS Enable**	
	Format:	Enable
	If set, PS reads color phases on per sample basis for each slot.	
	Programming Notes	
	This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE. When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0. When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples). When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.	
12	**Reserved**	
	Format:	MBZ
	Ignored	
11	**Slot Group Select**	

MSD_RTR_SIMD16 - SIMD16 Render Target Read MSD

		Format: MDC_RT_SGS This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
	10:9	**Reserved** Format: MBZ Ignored
	8	**Render Target Message Subtype** Default Value: 0h Format: Opcode SIMD16 single source message. Use slots [15:0] for the pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [31:16] are referenced instead of [15:0].
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

SIMD16 Render Target Write MSD

MSD_RTW_SIMD16 - SIMD16 Render Target Write MSD

Source: EuSubFunctionRenderDataPort
Length Bias: 1
Family: Other
Group: Render Target R/W

DWord	Bit	Description
0	31	**Reserved** Format: MBZ Ignored
	30	**Message Precision Subtype** Default Value: 0h Format: Opcode Full precision data message
	29	**Reserved** Format: MBZ Ignored
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present**

		Format: MDC_MHP If set, indicates that the message includes the 2-register header.
	18	**Per-Coarse Pixel PS outputs enable** Format: Enable This bit indicates the render target write is a coarse pixel write. **Programming Notes** This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor. When this bit is set and the message has oMask present, oMask represents the pixel enables within the Coarse Pixel in the row major order.
	17:14	**Message Type** Default Value: 0Ch Format: Opcode Render Target Write message
	13	**Per-Sample PS Enable** Format: Enable If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot. **Programming Notes** This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE. When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0. When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples). When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.
	12	**Last Render Target Select**

MSD_RTW_SIMD16 - SIMD16 Render Target Write MSD

		Format: Enable This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.
	11	**Slot Group Select** Format: MDC_RT_SGS This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
	10:8	**Render Target Message Subtype** Default Value: 0h Format: Opcode SIMD16 Single source message. Use slots [15:0] for pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [31:16] are referenced instead of [15:0].
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Split Send Message

sends - Split Send Message

Source: Eulsa

Length Bias: 4

Description
The sends instruction performs data communication between a thread and external function units, including shared functions (Sampler, Data Port Read, Data Port Write, URB, and Message Gateway) and some fixed functions (e.g. Thread Spawner, who also have an unique Shared Function ID). The sends instruction adds an entry to the EU's message request queue. The request message is stored in a block of contiguous GRF registers. The response message, if present, will be returned to a block of contiguous GRF registers. The return GRF writes may be in any order depending on the external function units. <src0> and <src1> are the lead GRF registers for the first and second block of the request respectively. <dest> is the lead GRF register for response. The message descriptor field <desc> contains the Message Length (the number of consecutive GRF registers corresponding to src0) and the Response Length (the number of consecutive GRF registers). It also contains the header present bit, and the function control signals. The extend message descriptor field <ex_desc> contains the target function ID, the Extended Message Length (the number of consecutive GRF registers corresponding to src1) and the extended function control signals. WrEn is forwarded to the target function in the message sideband. The sends instruction is the only way to terminate a thread. When the EOT (End of Thread) bit of <ex_desc> is set, it indicates the end of thread to the EU, the Thread Dispatcher and, in most cases, the parent fixed function.
Message descriptor field <desc> can be a 32-bit immediate, imm32, or a 32-bit scalar register, <reg32a>. GEN restricts that the 32-bit scalar register <reg32a> must be the leading dword of the address register. It should be in the form of a0.0<0;1,0>:ud. When <desc> is a register operand, only the lower 31 bits of <reg32a> are used.
Extended Message descriptor field <ex_desc> can be a 32-bit immediate, imm32 or a 32bit scalar register, <reg32a>. The bits3:0 of the <ex_desc> specifies the SFID for the message. The EOT field always comes from bit127 of the instruction word, which is the bit5 of <ex_desc>. A thread must terminate with a sends instruction with EOT turned on. The bits9:6 of <ex_desc> specify the extended message length and bits31:12 specify the 20bit extended function control. Interpretation of the extended function control signals is subject to the target external function. The scalar register <reg32a> is selected when SelReg32ExDesc is set, ExDesc.RegNum[3:0] provides the addressing for reg32a for extended message descriptor. This selects one of the index sub registers. Subregisters selected are always aligned to dword. This implies, the even index subregisters must be used.</reg32a></ex_desc></ex_desc></ex_desc></reg32a></ex_desc>
Function control is now extended to 20 bits as specified in the below definition.
Function control is now extended to 20 bits as specified in the below definition. The sum of Message Length and Extended Message Length must not be greater than 31.
<src0> is a 256-bit aligned GRF register. It serves as the leading GRF register of the request. <src1> is a 256-bit aligned GRF register or a null register. It serves as the leading GRF register for the second block of the request when it is not a null register. It is required that the second block of GRFs does not overlap with the first block. If it is a null register the Extended Message Length must be 0. The sum of Message Length

sends - Split Send Message

and Extended Message Length must not be greater than 15.
The source dependency control, {NoSrcDepSet} is used to control the setting of source dependency for both the sources.
<dest> serves for two purposes: to provide the leading GRF register location for the response message if present, and to provide parameters to form the channel enable sideband signals.
<dest> signals whether there is a response to the message request. It can be either a null register, a direct-addressed GRF register or a register-indirect GRF register. Otherwise, hardware behavior is undefined.
If <dest> is null, there is no response to the request. Meanwhile, the Response Length field in <desc> must be 0. Certain types of message requests, such as memory write (store) through the Data Port, do not want response data from the function unit. If so, the posted destination operand can be null.
If <dest> is a GRF register, the register number is forwarded to the shared function. In this case, the target function unit must send one or more response message phases back to the requesting thread. The number of response message phases must match the Response Length field in <desc>, which of course cannot be zero. For some cases, it could be an empty return message. An empty return message is defined as a single phase message with all channel enables turned off.
The destination type field is always valid and is used to generate the WrEn. This is true even if <dest> is a null register (this is an exception for null as for most cases these fields are ignored by hardware).
The address immediates for indirect sources and destination must be oword aligned.
The 16-bit channel enables of the message sideband are formed based on the WrEn. Interpretation of the channel enable sideband signals is subject to the target external function. In general for a 'sends' instruction with return messages, they are used as the destination dword write mask for the GRF registers starting at <dest>. For a message that has multiple return phases, the same set of channel enable signals applies to all the return phases.
NoDDClr and NoDDChk must not be used for send instruction.

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

Format:

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

Restriction

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

sends - Split Send Message

Syntax

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

Pseudocode

```
Evaluate(WrEn);
         <MsgChEnable> = WrEn;
         <SourceReg0> = <src0>.RegNum;
         <SourceReg1> = <src1>.RegNum;
         MessageEnqueue(<MsgChEnable>, <ResponseReg>, <SourceReg0>,
<SourceReg1>,<ex_desc>, <dest>);
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

DWord	Bit	Description
0..3	127:96	**Message** Format: EU_INSTRUCTION_OPERAND_SEND_MSG
	95:80	**ExDesc[31:16]** Format: ExtMsgDescpt[31:16]
	79	**Source 0 Addressing Mode** Format: AddrMode
	78	**Reserved** Exists If: ([Source 0 Addressing Mode]=='Direct') Format: MBZ
	78	**Source 0 Address Immediate Sign [9]** Exists If: ([Source 0 Addressing Mode]=='Indirect') Format: S9[9]
	77	**SelReg32Desc** Indicate the source of Message Descriptor. Immediate value from instruction of indirect value from address registers. Value: 0 — Name: IMM Value: 1 — Name: REG32
	76:73	**Source 0 Address Subregister Number** Exists If: ([Source 0 Addressing Mode]=='Indirect')
	76:69	**Source 0 Register Number** Exists If: ([Source 0 Addressing Mode]=='Direct')
	72:68	**Source 0 Address Immediate [8:4]**

sends - Split Send Message

		Exists If: ([Source 0 Addressing Mode]=='Indirect') Format: S9[8:4]
	68	**Source 0 Subregister Number** Exists If: ([Source 0 Addressing Mode]=='Direct')
	67:64	**ExDesc[9:6]** Format: **ExtMsgDescpt[9:6]**
	63	**Destination Addressing Mode** Format: **AddrMode**
	62	**Destination Address Immediate Sign [9]** Exists If: ([Destination Addressing Mode]=='Indirect') Format: S9[9]
	62	**Reserved** Exists If: ([Destination Addressing Mode]=='Direct') Format: MBZ
	61	**SelReg32ExDesc** Indicate the source of Extended Message Descriptor. Immediate value from instruction of indirect value from address registers. Value: Name 0: IMM 1: REG32
	60:57	**Destination Address Subregister Number** Exists If: ([Destination Addressing Mode]=='Indirect')
	60:53	**Destination Register Number** Exists If: ([Destination Addressing Mode]=='Direct')
	56:52	**Destination Address Immediate [8:4]** Exists If: ([Destination Addressing Mode]=='Indirect') Format: S9[8:4]
	52	**Destination Subregister Number [4]** Exists If: ([Destination Addressing Mode]=='Direct')
	51:44	**Source 1 Register Number**
	43:41	**Reserved** Format: MBZ
	40:37	**Destination Type**
	36	**Source 1 Register File** Format: **RegFile[0]**

sends - Split Send Message

	35	**Destination Register File** Format: RegFile[0]
	34	**MaskCtrl**
	33:32	**Flag Register Number/Subregister Number**
	31:28	**Controls B** Format: EU_INSTRUCTION_CONTROLS_B
	27:24	**Shared Function ID (SFID)** Format: SFID
	23:8	**Controls A** Format: EU_INSTRUCTION_CONTROLS_A
	7	**Reserved** Format: MBZ
	6:0	**Opcode** Format: EU_OPCODE

STATE_BASE_ADDRESS

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.
For more information see the Base Address Utilization table in the Memory Access Indirection narrative topic.

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

Execution of this command causes a full pipeline flush, thus its use should be minimized for higher performance.

SW must always program PIPE_CONTROL with "CS Stall" and "Render Target Cache Flush Enable" set before programming STATE_BASE_ADDRESS command for GPGPU workloads i.e when pipeline select is GPGPU via PIPELINE_SELECT command. This is required to achieve better GPGPU preemption latencies in certain workload programming sequences. If programming PIPE_CONTROL has performance implications then preemption latencies can be traded off against performance by not implementing this programming note.

SW must always program PIPE_CONTROL with “Command Cache Invalidate Enable” following programming of STATE_BASE_ADDRESS command when “State Cache redirect to CS Section enable” bit is set in MMIO register SLICE_COMMON_ECO_CHCICKEN1 (0731Ch).

SW must always program MEDIA_STATE_FLUH command with “Flush To GO” set prior to programming of STATE_BASE_ADDRESS command for GPGPU/Media workloads i.e when pipeline select is GPGPU or Media via PIPELINE_SELECT command. This is required to ensure the write data out of the prior thread group are flushed out prior to the state changes due to the programming of STATE_BASE_ADDRESS command take place.

DWord	Bit	Description	
0	31:29	**Command Type**	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	**Command SubType**	
		Default Value:	0h GFXPIPE_COMMON
		Format:	OpCode
	26:24	**3D Command Opcode**	
		Default Value:	1h GFXPIPE_NONPIPELINED

		Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 01h STATE_BASE_ADDRESS Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2 **Value** / **Name** / **Description** 14h / DWORD_COUNT_n **[Default]** / Excludes DWord (0,1)
1..2	63:12	**General State Base Address** Format: GraphicsAddress[63:12] Specifies the 4K-byte aligned base address for general state accesses. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47]. **Programming Notes** Bounds checking is performed on general state accesses by Data Port Shared Functions for stateless A32 messages. Bounds checking is enabled when General State Base Address [46:12] + General State Buffer Size [31:12] is <= 2^47. This ensures that the General State Buffer does not straddle the canonical address boundary where GraphicsAddress [47] changes. **Restriction** General State Base Address [47:12] + General State Buffer Size [31:12] must be < 2^48. It is illegal programming for this to be >= 2^48. When using stateless (A32) Data Port messages, General State Base Address [47:12] + Buffer Base Address [31:0] must be < 2^48. It is illegal for this to be >= 2^48.
	11	**Reserved** Format: MBZ
	10:4	**General State Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for indirect state using the **General State Base Address**, with the exception of the stateless data port accesses.

STATE_BASE_ADDRESS

	3:1	**Reserved** Format: MBZ
	0	**General State Base Address Modify Enable** Format: Enable The other fields in this DWord and the following DWord are updated only when this bit is set. 0h – Disable – Ignore the updated address. 1h – Enable – Modify the address.
3	31:23	**Reserved** Format: MBZ
	22:16	**Stateless Data Port Access Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for stateless data port accesses.
	15:13	**Reserved** Format: MBZ
	12:1	**Reserved** Format: MBZ
	0	**Reserved** Format: MBZ
4..5	63:12	**Surface State Base Address** Format: GraphicsAddress[63:12] Specifies the 4K-byte aligned base address for binding table and surface state accesses. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].
	11	**Reserved**

		Format: MBZ
	10:4	**Surface State Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for indirect state using the **Surface State Base Address**.
	3:1	**Reserved** Format: MBZ
	0	**Surface State Base Address Modify Enable** Format: Enable The other fields in this DWord and the following DWord are updated only when this bit is set.

Value	Name	Description
0h	Disable	Ignore the updated address.
1h	Enable	Modify the address.

Programming Notes
Setting this bit to 1 in a batch buffer causes the resource streamer to stop; for performance reasons the SW should only place commands with this bit set in the ring buffer.
Before programming the Surface State Base Address, the RS must be disabled. Within a batch buffer where the RS is enabled, RS may be disabled thru a MI_RS_CONTROL command with Resource Streamer Control cleared prior to the STATE_BASE_ADDRESS with Surface State Base Address Modify Enable set and then re-enabled with another MI_RS_CONTROL with Resource Streamer Control set.

6..7	63:12	**Dynamic State Base Address** Format: GraphicsAddress[63:12] Specifies the 4K-byte aligned base address for sampler and viewport state accesses. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].
	11	**Reserved** Format: MBZ
	10:4	**Dynamic State Memory Object Control State**

		Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for indirect state using the **Dynamic State Base Address**. Push constants defined in 3DSTATE_CONSTANT_(VS | GS | PS) commands do not use this control state, although they can use the corresponding base address. The memory object control state for push constants is defined within the command.
	3:1	**Reserved** Format: MBZ
	0	**Dynamic State Base Address Modify Enable** Format: Enable The other fields in this DWord and the following DWord are updated only when this bit is set. **Value** / **Name** / **Description** 0h / Disable / Ignore the updated address. 1h / Enable / Modify the address.
8..9	63:12	**Indirect Object Base Address** Format: GraphicsAddress[63:12]IndirectObject Specifies the 4K-byte aligned base address for indirect object load in MEDIA_OBJECT command.
	11	**Reserved** Format: MBZ
	10:4	**Indirect Object Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for indirect objects using the **Indirect Object Base Address**.
	3:1	**Reserved** Format: MBZ
	0	**Indirect Object Base Address Modify Enable** Format: Enable The other fields in this DWord and the following DWord are updated only when this bit is set. **Value** / **Name** / **Description**

		0h	Disable	Ignore the updated address.
		1h	Enable	Modify the address.
10..11	63:12	**Instruction Base Address**		
		Format:	GraphicsAddress[63:12]	
		Specifies the 4K-byte aligned base address for all EU instruction accesses. GraphicsAddress[63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].		
	11	**Reserved**		
		Format:	MBZ	
	10:4	**Instruction Memory Object Control State**		
		Format:	MEMORY_OBJECT_CONTROL_STATE	
		Specifies the memory object control state for EU instructions using the **Instruction Base Address**.		
	3:1	**Reserved**		
		Format:	MBZ	
	0	**Instruction Base Address Modify Enable**		
		Format:	Enable	
		The other fields in this DWord and the following DWord are updated only when this bit is set.		
		Value	**Name**	**Description**
		0h	Disable	Ignore the updated address.
		1h	Enable	Modify the address.
12	31:12	**General State Buffer Size**		
		Format:	U20	
		FormatDesc		
		This field specifies the size of the buffer in 4K pages. Any access that straddles or goes past the end of the buffer returns 0. Note that BufferSize=0 indicates that there is no valid data in the buffer.		
	11:1	**Reserved**		

		Format: MBZ
	0	**General State Buffer Size Modify Enable** Format: Enable The bound in this DWord is updated only when this bit is set. 0h – Disable – Ignore the updated bound. 1h – Enable – Modify the updated bound.
13	31:12	**Dynamic State Buffer Size** Format: U20 FormatDesc This field specifies the size of the buffer in 4K pages. Any access that straddles or goes past the end of the buffer returns 0. Note that BufferSize=0 indicates that there is no valid data in the buffer.
	11:1	**Reserved** Format: MBZ
	0	**Dynamic State Buffer Size Modify Enable** Format: Enable FormatDesc The bound in this DWord is updated only when this bit is set. 0h – Disable – Ignore the updated bound. 1h – Enable – Modify the updated bound.
14	31:12	**Indirect Object Buffer Size** Format: U20 FormatDesc This field specifies the size of the buffer in 4K pages. Any access that straddles or goes past the end of the buffer returns 0.

		Note that BufferSize=0 indicates that there is no valid data in the buffer.
	11:1	**Reserved** Format: MBZ
	0	**Indirect Object Buffer Size Modify Enable** Format: Enable FormatDesc: The bound in this DWord is updated only when this bit is set. 0h Disable: Ignore the updated bound. 1h Enable: Modify the updated bound.
15	31:12	**Instruction Buffer Size** Format: U20 FormatDesc: This field specifies the size of the buffer in 4K pages. Any access that straddles or goes past the end of the buffer returns 0. Note that BufferSize=0 indicates that there is no valid data in the buffer.
	11:1	**Reserved** Format: MBZ
	0	**Instruction Buffer size Modify Enable** Format: Enable FormatDesc: The bound in this DWord is updated only when this bit is set. 0h Disable: Ignore the updated bound.
16..17	63:12	**Bindless Surface State Base Address** Format: GraphicsAddress[63:12]

Specifies the 4K-byte aligned base address for bindless surface state accesses. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].

11 — Reserved

Format:	MBZ

10:4 — Bindless Surface State Memory Object Control State

Format:	MEMORY_OBJECT_CONTROL_STATE

Specifies the memory object control state for indirect state using the **Bindless Surface State Base Address**.

3:1 — Reserved

Format:	MBZ

0 — Bindless Surface State Base Address Modify Enable

Format:	Enable

Description
The other fields in this DWord and the following two DWords are updated only when this bit is set.

Value	Name	Description
0h	Disable	Ignore the updated address
1h	Enable	Modify the address

18 — 31:12 — Bindless Surface State Size

Format:	U20

This field indicates the size-1 of the Bindless Surface State buffer in 64-Byte increments. Any SSO beyond this maximum size points to offset 0.
Example: If the buffer contains 512 surface states, then this field must be programmed to 0x1FF (511 decimal).

11:0 — Reserved

Format:	MBZ

19..20 — 63:12 — Bindless Sampler State Base Address

		Format: GraphicsAddress[63:12] Specifies the 4K-byte aligned base address for bindless sampler state accesses. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].
	11	**Reserved** Format: MBZ
	10:4	**Bindless Sampler State Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for indirect state using the **Bindless Sampler State Base Address**.
	3:1	**Reserved** Format: MBZ
	0	**Bindless Sampler State Base Address Modify Enable** Format: Enable The other fields in this DWord and the following two DWords are updated only when this bit is set.

Value	Name	Description
0h	Disable	Ignore the updated address
1h	Enable	Modify the address

21	31:12	**Bindless Sampler State Buffer Size** Format: U20 This field specifies the size of the buffer in 4K pages. Any access that goes beyond the end of the buffer (as defined by the Sampler State Buffer Size) will use an offset of 0.
	11:0	**Reserved** Format: MBZ

STATE_SIP

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.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 0h GFXPIPE_COMMON Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h GFXPIPE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 02h STATE_SIP Format: OpCode
	15:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2 Value / Name / Description: 1h / DWORD_COUNT_n **[Default]** / Excludes DWord (0,1)
1..2	63:4	**System Instruction Pointer** Format: InstructionBaseOffset[63:4]Kernel Specifies the instruction address of the system routine associated with the current context as a 128-bit granular offset from the Instruction Base Address. SIP is shared by all threads in execution. The address specifies the double quadword aligned instruction location. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form

STATE_SIP		
		[63:48] == [47]. **Programming Notes** This portion of the command is not context save/restored. The context image may restore this command as a 2 dword command rather than a 3 dword command.
	3:0	**Reserved** Format: MBZ

Sum of Absolute Difference 2

sad2 - Sum of Absolute Difference 2

Source: Eulsa

Length Bias: 4

The sad2 instruction takes source data channels from src0 and src1 in groups of 2-tuples. For each 2-tuple, it computes the sum-of-absolute-difference (SAD) between src0 and src1 and stores the scalar result in the first channel of the 2-tuple in dst. The results are also stored in the accumulator register. The destination operand and the accumulator maintain 16 bits per channel precision. The destination register must be aligned to even word (DWord). The even words in the destination region will contain the correct data. The odd words are also written but with undefined values.

Format:

```
[(pred)] sad2[.cmod] (exec_size) dst src0 src1
```

Restriction

Source operands cannot be accumulators.

The execution size cannot be 1 as the computation requires at least two data channels.

Syntax

```
[(pred)] sad2[.cmod] (exec_size) reg reg reg
[(pred)] sad2[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
     for ( n = 0; n < exec_size; n += 2 ) {
         if ( WrEn.chan[n] ) {
            dst.chan[n] = abs(src0.chan[n] - src1.chan[n])
                        + abs(src0.chan[n+1] - src1.chan[n+1]);
         }
     }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types
B, UB	W, UW

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

sad2 - Sum of Absolute Difference 2		
		Exists If: ([ImmSource][Src1.RegFile]=='IMM') Format: **EU_INSTRUCTION_SOURCES_REG_IMM**
	63:32	**Operand Controls** Format: **EU_INSTRUCTION_OPERAND_CONTROLS**
	31:0	**Header** Format: **EU_INSTRUCTION_HEADER**

Sum of Absolute Difference Accumulate 2

sada2 - Sum of Absolute Difference Accumulate 2

Source: Eulsa

Length Bias: 4

The sada2 instruction takes source data channels from src0 and src1 in groups of 2-tuples. For each 2-tuple, it computes the sum-of-absolute-difference (SAD) between src0 and src1, adds the intermediate result with the accumulator value corresponding to the first channel, and stores the scalar result in the first channel of the 2-tuple in dst. The destination operand and the accumulator maintain 16 bits per channel precision. Higher precision (guide bits) stored in the accumulator allows up to 64 rounds of sada2 instructions to be issued back to back without overflowing the accumulator. The destination register must be aligned to even word (DWord). The even words in the destination region will contain the correct data. The odd words are also written but with undefined values.

Format:

```
[(pred)] sada2[.cmod] (exec_size) dst src0 src1
```

Restriction

Source operands cannot be accumulators.

The execution size cannot be 1 as the computation requires at least two data channels.

Syntax

```
[(pred)] sada2[.cmod] (exec_size) reg reg reg
[(pred)] sada2[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
   for ( n = 0; n < exec_size; n += 2 ) {
      uwTmp = abs(src0.chan[n] - src1.chan[n])
            + abs(src0.chan[n+1] - src1.chan[n+1]);
      if ( WrEn.chan[n] ) {
         dst.chan[n] = uwTmp + acc[n];
      }
   }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

Src Types	Dst Types
B, UB	W, UW

DWord	Bit	Description
0..3	127:64	**RegSource** Exists If: ([RegSource][Src1.RegFile]!='IMM')

sada2 - Sum of Absolute Difference Accumulate 2

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

Typed Surface Read MSD

MSD1R_TS - Typed Surface Read MSD

Source:	EuSubFunctionDataPort1
Length Bias:	1
Family:	Typed Surface R/W
Group:	Scattered Typed Surface R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 05h Format: Opcode Typed Surface Read message
	13:12	**Slot Group** Format: MDC_SG3 Specifies the Slot Group mode of the message (which slots are processed)
	11:8	**Channel Mask** Format: MDC_CMASK Specifies which RGBA channels are included in the message payload.
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Typed Surface Write MSD

MSD1W_TS - Typed Surface Write MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Typed Surface R/W
Group: Scattered Typed Surface R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 0Dh Format: Opcode Typed Surface Write message
	13:12	**Slot Group** Format: MDC_SG3

<table>
<tr><th colspan="3">MSD1W_TS - Typed Surface Write MSD</th></tr>
<tr><td></td><td></td><td>Specifies the Slot Group mode of the message (which slots are processed)</td></tr>
<tr><td></td><td>11:8</td><td>Channel Mask<table><tr><td></td><td></td></tr><tr><td>Format:</td><td>MDC_CMASK</td></tr></table>Specifies which RGBA channels are included in the message payload.</td></tr>
<tr><td></td><td>7:0</td><td>Binding Table Index<table><tr><td></td><td></td></tr><tr><td>Format:</td><td>MDC_BTS</td></tr></table>Specifies the Binding Table Index for the message</td></tr>
</table>

Untyped Surface Read MSD

MSD1R_US - Untyped Surface Read MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Surface R/W
Group: Scattered Untyped Surface R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 01h Format: Opcode Untyped Surface Read message
	13:12	**SIMD Mode** Format: MDC_SM3

<table>
<tr><th colspan="3">MSD1R_US - Untyped Surface Read MSD</th></tr>
<tr><td></td><td></td><td>Specifies the SIMD mode of the message (number of slots processed)</td></tr>
<tr><td></td><td>11:8</td><td>Channel Mask<table><tr><td></td><td></td></tr><tr><td>Format:</td><td>MDC_CMASK</td></tr></table>Specifies which RGBA channels are included in the message payload.</td></tr>
<tr><td></td><td>7:0</td><td>Binding Table Index<table><tr><td></td><td></td></tr><tr><td>Format:</td><td>MDC_BTS_SLM_A32</td></tr></table>Specifies the Binding Table Index for the message</td></tr>
</table>

Untyped Surface Write MSD

MSD1W_US - Untyped Surface Write MSD

Source: EuSubFunctionDataPort1
Length Bias: 1
Family: Untyped Surface R/W
Group: Scattered Untyped Surface R/W

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	**Message Type** Default Value: 09h Format: Opcode Untyped Surface Write message
	13:12	**SIMD Mode** Format: MDC_SM3

<table>
<tr><th colspan="3">MSD1W_US - Untyped Surface Write MSD</th></tr>
<tr><td></td><td></td><td>Specifies the SIMD mode of the message (number of slots processed)</td></tr>
<tr><td></td><td>11:8</td><td>Channel Mask<table><tr><td></td><td></td></tr><tr><td>Format:</td><td>MDC_UW_CMASK</td></tr></table>Specifies which RGBA channels are included in the message payload.</td></tr>
<tr><td></td><td>7:0</td><td>Binding Table Index<table><tr><td></td><td></td></tr><tr><td>Format:</td><td>MDC_BTS_SLM_A32</td></tr></table>Specifies the Binding Table Index for the message</td></tr>
</table>

URB Dword Read MSD

MSDUR_DWS - URB Dword Read MSD

Source: EuSubFunctionReadOnlyDataPort

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Default Value: 1 Present Format: Opcode Indicates that the message requires a header.
	18	**Reserved** Format: MBZ
	17	**Per Slot Offset Present** Format: Enable Specifies if per-slot offset message payload is present.
	16	**Reserved** Format: MBZ
	15	**Channel Mask Present** Default Value: 0 Not Present Format: Opcode Must be clear on read messages, indicating the Channel Mask Message phase is not present.
	14:4	**Global Offset**

MSDUR_DWS - URB Dword Read MSD

		Format: U11 Specifies the offset, in units of Oword elements, from the start of the URB handle for the access. If **Per Slot Offset Present** is set, this global offset is added to each of the slot offsets to form the overall offset. **Value** / **Name** [0-2047]
	3:0	**URB Opcode** Format: Opcode **Value** / **Name** / **Description** 8 / URB_SIMD8_READ **[Default]** / SIMD8 URB Dword Read message. Reads 1..8 Dwords, based on RLEN.

URB Dword Write MSD

MSDUW_DWS - URB Dword Write MSD

Source: EuSubFunctionReadOnlyDataPort

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Default Value: 1 Present Format: Opcode Indicates that the message requires a header.
	18	**Reserved** Format: MBZ
	17	**Per Slot Offset Present** Format: Enable Specifies if per-slot offset message payload is present. If present, it will be added to the **Global Offset**.
	16	**Reserved** Format: MBZ
	15	**Channel Mask Present** Default Value: 0 Not Present Format: Opcode Indicates the channel Mask Message phase is not present.
	14:4	**Global Offset**

MSDUW_DWS - URB Dword Write MSD

		Format: U11 Specifies the offset, in units of Oword elements, from the start of the URB handle for the access. If **Per Slot Offset** is set, the global offset is added to those offsets to form the overall offset. **Value** / **Name** [0-2047]
	3:0	**URB Opcode** Format: Opcode **Value** / **Name** / **Description** 7 / URB_SIMD8_WRITE **[Default]** / SIMD8 URB Dword Write message. Writes1..8 Dwords, based on RLEN and Channel Mask.

URB Masked Dword Write MSD

MSDUW_MDWS - URB Masked Dword Write MSD

Source: EuSubFunctionReadOnlyDataPort

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Default Value: 1 Present Format: Opcode Indicates that the message requires a header.
	18	**Reserved** Format: MBZ
	17	**Per Slot Offset Present** Format: Enable Specifies if per-slot offset message payload is present. If present, it will be added to the **Global Offset**.
	16	**Reserved** Format: MBZ
	15	**Channel Mask Present** Default Value: 1 Present Format: Opcode Indicates the Channel Mask Message phase is present and will be used to mask which data elements written.

MSDUW_MDWS - URB Masked Dword Write MSD

	14:4	**Global Offset** Format: U11 Specifies the offset, in units of Oword elements, from the start of the URB handle for the access. If **Per Slot Offset** is set, the global offset is added to those offsets to form the overall offset. **Value** / **Name** [0-2047]
	3:0	**URB Opcode** Format: Opcode **Value** / **Name** / **Description** 7 / URB_SIMD8_WRITE **[Default]** / SIMD8 URB Dword Write message. Writes1..8 Dwords, based on RLEN and Channel Mask.

VD_PIPELINE_FLUSH

VD_PIPELINE_FLUSH

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 Media Format: OpCode
	26:23	**Media Command Opcode** Default Value: Fh Extended command Format: OpCode
	22:21	**SubOpcodeA** Default Value: 0h Format: OpCode
	20:16	**SubOpcodeB** Default Value: 0h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWORD_COUNT_n** Default Value: 0h Excludes DWord (0) Format: =n Total Length - 2
1	31:21	**Reserved** Format: MBZ
	20	**Reserved** Format: MBZ

VD_PIPELINE_FLUSH		
	19	**MFX pipeline command flush** Format: U1
	18	**Reserved**
	17	**VD-ENC pipeline command flush** Format: U1
	16	**HEVC pipeline command flush** Format: U1
	15:6	**Reserved** Format: MBZ
	5	**Reserved** Format: MBZ
	4	**VD command/message parser Done** Format: U1
	3	**MFX pipeline Done** Format: U1
	2	**Reserved**
	1	**VD-ENC pipeline Done** Format: U1
	0	**HEVC pipeline Done** Format: U1

VDENC_CONST_QPT_STATE

VDENC_CONST_QPT_STATE

Source: VideoCS

Length Bias: 2

This commands provides the tables for frame constants to the VDEnc HW. The specific parameter value is picked by the VDEnc HW based on the frame level QP. The QP Lambda array for costing (motion-vectors and mode costs) has 42 entries. Skip Threshold tables has 27 entries. 7 FTQ thresholds [0-6] are programmed using 4 sets of tables with 27 entries each.

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
	26:23	**Opcode** Default Value: 1h VDENC_PIPE Format: OpCode
	22:21	**SubOpA** Default Value: 0h Format: OpCode
	20:16	**SubOpB** Default Value: 6h VDENC_CONST_QPT_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Total Length - 2 Value: 3Ch — Name: DWORD_COUNT_n **[Default]** — Description: Excludes DWord (0,1)
1..11	351:336	**Reserved** Format: MBZ
	335:0	**QP Lambda Array Index[n]**

		Format: U8[42] The distance between the current MV and PMV is multiplied by a QP based lambda value to get the mvcost. The QP index for this array is generated by the following equation : max(0, (QpPrimeY - 10)).
12..25	447:432	**Reserved** Format: MBZ
	431:0	**Skip Threshold Array Index[n]** Format: U16[27] This value is used for SkipCheck if SkipFTQ is disabled. This value should be programmed to be compliant with the block size being checked for determining skip success in the HW. It could be 16x16, 8x8 or 4x4 based. In the case, FTQ is enabled, the hardware checks for non-zero coefficients on a per 8x8 block basis to determine if the MB can be coded as a SKIP or not. The VDEnc only provides a hint to the PAK. The actual SKIPMB decision is performed in the PAK. The QP index for this array is generated by the following equation - ((QpPrimeY + 1) » 1).
26..39	447:432	**Reserved** Format: MBZ
	431:0	**SIC Forward Transform Coeff Threshold Matrix0 Array Index[n]** Format: U16[27] Values of the threshold matrix 0 is provided here. Matrix 0 contains the DC threshold for the Forward Transform Skip check. It has increased precision (Format = U16) vs the other thresholds due to the larger size of DC coeffieicients. Matrix 1 through Matrix 6 have lower precision (Format = U8). HW internally left shifts the AC coefficients (Matrix 1 to 6) by 2 to get to a 10-bit range if the frame level QP (QpPrimeY) is >= 37. The QP index for this array is generated by the following equation - ((QpPrimeY + 1) » 1). Threshold Matrix for 4x4 transform is as follows: 0 1 2 3 1 4 3 4 2 3 2 5 3 4 5 6
40..46	223:216	**Reserved** Format: MBZ
	215:0	**SIC Forward Transform Coeff Threshold Matrix1/3/5 Array Index[n]** Format: U8[27] Values of the threshold matrix 1, 3 and 5 are provided here.

VDENC_CONST_QPT_STATE		
47..53	223:216	**Reserved** Format: MBZ
	215:0	**SIC Forward Transform Coeff Threshold Matrix2 Array Index[n]** Format: U8[27] Values of the threshold matrix 2 are provided here.
54..60	223:216	**Reserved** Format: MBZ
	215:0	**SIC Forward Transform Coeff Threshold Matrix4/6 Array Index[n]** Format: U8[27] Values of the threshold matrix 4 and 6 are provided here.

VDENC_DS_REF_SURFACE_STATE

VDENC_DS_REF_SURFACE_STATE

Source: VideoCS

Length Bias: 2

This command specifies the surface state parameters for the downscaled reference surfaces.

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
	26:23	**Opcode** Default Value: 1h VDENC_PIPE Format: OpCode
	22:21	**SubOpA** Default Value: 0h Format: OpCode
	20:16	**SubOpB** Default Value: 3h VDENC_DS_REF_SURFACE_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Total Length - 2 Value: 4h — Name: DWORD_COUNT_n **[Default]** — Description: Excludes DWord (0,1)
1	31:4	**Reserved** Format: MBZ Surface Base Address is NOT used for codec H/W. This field is reserved for 3D surface state compatibility. VDEnc pipeline gets this address from VDENC_PIPE_BUF_ADDR_STATE for different buffers.
	3:0	**Reserved**

VDENC_DS_REF_SURFACE_STATE			
		Format:	MBZ
2..5	127:0	**Dwords 2..5** Format: VDENC_Surface_State_Fields Four DWords that define the bit fields used by the three surface state commands.	

VDENC_PIPE_BUF_ADDR_STATE

VDENC_PIPE_BUF_ADDR_STATE

Source: VideoCS

Length Bias: 2

This state command provides the memory base addresses for all row stores, Streamin/StreamOut, DMV buffer along with the uncompressed source, reference pictures and downscaled reference pictures required by the VDENC pipeline. All reference pixel surfaces in the Encoder are programmed with the same surface state (NV12 and TileY format), except each has its own frame buffer base address. Same holds true for the down-scaled reference pictures too. 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. VDEnc supports 3 Downscaled reference frames (2 fwd, 1 bwd) and 4 normal reference frames (3 fwd, 1 bwd). The driver will sort out the base address from the DPB table and populate the base addresses that map to the corresponding reference index for both DS references and normal reference frames.
Each of the individual DS ref/ Normal ref frames have their own MOCS DW that corresponds to the respective base address. The only thing that is different in the MOCS DW amongst the DS reference frames is the MMCD controls (specified in bits [10:9] of the MOCS DW). Driver needs to ensure that the other bits need to be the same across the different DS ref frames. The same is applicable for the normal reference frames.

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
	26:23	**Opcode** Default Value: 1h VDENC_PIPE Format: OpCode
	22:21	**SubOpA** Default Value: 0h Format: OpCode
	20:16	**SubOpB** Default Value: 4h VDENC_PIPE_BUF_ADDR_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length**

VDENC_PIPE_BUF_ADDR_STATE

		Format: =n Total Length - 2 **Value** / **Name** / **Description** 24h / DWORD_COUNT_n **[Default]** / Excludes DWord (0,1)
1..3	95:0	**DS FWD REF0** Format: VDENC_Down_Scaled_Reference_Picture
4..6	95:0	**DS FWD REF1** Format: VDENC_Down_Scaled_Reference_Picture **Description** AVC Mode: This field must be programmed to have the same base address as DS FWD REF0 [List 0, refidx 0] when VDEncPerfMode is enabled in the VDENC_IMAGE_STATE. In Normal Mode, This field points to DS FWD REF1 [List 0, refidx 1]
7..9	95:0	**Reserved** Format: MBZ
10..12	95:0	**Original Uncompressed Picture** Format: VDENC_Original_Uncompressed_Picture
13..15	95:0	**Streamin Data Picture** Format: VDENC_Streamin_Data_Picture
16..18	95:0	**Row Store Scratch Buffer** Format: VDENC_Row_Store_Scratch_Buffer_Picture
19..21	95:0	**Colocated MV Read Buffer** Format: VDENC_Colocated_MV_Picture
22..24	95:0	**FWD REF0** Format: VDENC_Reference_Picture **Description** IN HEVC / VP9 mode the values in this field are ignored. The reference address programmed in PAK is used. **Programming Notes** The reference base address and cache control values programmed here would have to match reference picture address seen by PAK after HCP_REF_IDX translation. Example: Reference Picture 7 is used as reference for current frame. VDENC FWD REF0 should be programmed with [7] HCP_REF_IDX (L0) should map to reference picture 7 for reference index 0 in Reference

		picture list 0. This reference address points to current frame pre-deblocked reconstructed if NumRefIdxL0_minus1 equals 0 and TBC is enabled.
25..27	95:0	**FWD REF1** Format: VDENC_Reference_Picture **Description** IN HEVC / VP9 mode the values in this field are ignored. The reference address programmed in PAK is used. **Programming Notes** The reference base address and cache control values programmed here would have to match reference picture address seen by PAK after HCP_REF_IDX translation. Example: Reference Picture6 is used as reference for current frame. VDENC FWD REF1 should be programmed with [6] HCP_REF_IDX (L0) should map to reference picture6 for reference index1 in Reference picture list 0. This reference address points to current frame pre-deblocked reconstructed if NumRefIdxL0_minus1 equals 1 and TBC is enabled.
28..30	95:0	**FWD REF2** Format: VDENC_Reference_Picture **Description** IN HEVC / VP9 mode the values in this field are ignored. The reference address programmed in PAK is used. **Programming Notes** The reference base address and cache control values programmed here would have to match reference picture address seen by PAK after HCP_REF_IDX translation. Example: Reference Picture0 is used as reference for current frame. VDENC FWD REF1 should be programmed with [0] HCP_REF_IDX (L0) should map to reference picture0 for reference index2 in Reference picture list 0. This reference address points to current frame pre-deblocked reconstructed if NumRefIdxL0_minus1 equals2 and TBC is enabled.
31..33	95:0	**BWD REF0** Format: VDENC_Reference_Picture **Programming Notes** The reference base address and cache control values programmed here would have to match

VDENC_PIPE_BUF_ADDR_STATE		
		reference picture address seen by PAK after HCP_REF_IDX translation. Example: Reference Picture1 is used as only backward reference. VDENC BWD REF0 should be programmed with [1] HCP_REF_IDX (L1) should map to reference picture1 for reference index0 in Reference picture list 1. This reference address points to current frame pre-deblocked reconstructed if NumRefIdxL0_minus1 equals3 and TBC is enabled.
34..36	95:0	**VDEnc Statistics Streamout** Format: VDENC_Statistics_Streamout This parameter indicates the statistics streamout surface.
37..39	95:0	**DS FWD REF0 4X** Format: **VDENC_Down_Scaled_Reference_Picture** For HEVC VDEnc, this field indicates the 4X DS surface base address used in the stage2 ME for Reference frame0.
40..42	95:0	**DS FWD REF1 4X** Format: **VDENC_Down_Scaled_Reference_Picture**
43..45	95:0	**Reserved** Format: MBZ

VDENC_PIPE_MODE_SELECT

VDENC_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 **VDENC_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.

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
	26:23	**Opcode** Default Value: 1h VDENC_PIPE Format: OpCode
	22:21	**SubOpA** Default Value: 0h Format: OpCode
	20:16	**SubOpB** Default Value: 0h VDENC_PIPE_MODE_SELECT Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Total Length - 2 Value: 0h — Name: **[Default]** — Description: Excludes DWord (0,1)
1	30:28	**Reserved** Format: MBZ
	27:26	**Reserved**

25	**Reserved**	
24:20	**Reserved**	
19:18	**Reserved**	
17	**output range control after color space conversion** This bit allows control of swing on the output. Input is always full swing RGB. 0 - RGB to YUV using full to full 1 - RGB to YUV using full to studio range **Programming Notes** Full to studio range can be supported starting from Gen11.	
16:15	**PAK chroma sub-sampling type** Format: U2 This field is applicable only in HEVC and VP9. In AVC, this field is ignored. **Value / Name / Description** 0 / Reserved / 1 / 4:2:0 / Used for Main8 and Main10 HEVC, VP9 profile0. 3 / 4:4:4 / HEVC RExt 444, VP9 444 profiles.	
9	**VDEnc Stream-in Enable** Format: Enable This field controls whether VDEnc will read the stream-in surface that is programmed. Currently the stream-in surface has MB level QP, ROI, predictors and MaxSize/TargetSizeinWordsMB parameters. The individual enables for each of the fields is programmed in the VDENC_IMG_STATE. (ROI_Enable, Fwd/Predictor0 MV Enable, Bwd/Predictor1 MV Enable, MB Level QP Enable, TargetSizeinWordsMB/MaxSizeinWordsMB Enable). This bit is valid only in AVC mode. In HEVC / VP9 mode this bit is reserved and should be set to zero. **Value / Name** 0h / Disable 1h / Enable **Programming Notes** VDEnc stream-in can be enabled on a per frame basis. However it should only be enabled if there is valid data in the streamin surface (indicated by the individual enables in the Image State).	

VDENC_PIPE_MODE_SELECT

8 — PAK Threshold Check Enable

For AVC standard: This field controls whether VDEnc will check the PAK indicator for bits overflow and terminates the slice. This mode is called Dynamic Slice Mode. When this field is disabled, VDEnc is in Static Slice Mode. It uses the driver programmed Slice Macroblock Height Minus One to terminate the slice. This feature is also referred to as slice size conformance.
For HEVC standard: This bit is used to enable dynamic slice size control.

Value	Name
0h	Disable [Static Slice Mode]
1h	Enable [Dynamic Slice Mode]

Programming Notes
This feature can't be enabled if PAK only second pass support is needed. This field should be programmed as zero in vp9.

7 — TLB Prefetch Enable

Format:	Enable

This field controls whether TLB prefetching is enabled.

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

5 — Frame Statistics Stream-Out Enable

Format:	Enable

This field controls whether the frame statistics stream-out is enabled.

Value	Name
0h	Disable
1h	Enable

4 — Scalability Mode

Format:	Enable

When this is set, frame is encoded using multiple VDENC pipes. This bit is not used by HW.

3:0 — Standard Select

Format:	U4

Value	Name
2	AVC
3	Reserved
[4-15]	Reserved

VDENC_REF_SURFACE_STATE

VDENC_REF_SURFACE_STATE

Source: VideoCS

Length Bias: 2

This command specifies the surface state parameters for the normal reference surfaces.

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
	26:23	**Opcode** Default Value: 1h VDENC_PIPE Format: OpCode
	22:21	**SubOpA** Default Value: 0h Format: OpCode
	20:16	**SubOpB** Default Value: 2h VDENC_REF_SURFACE_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Total Length - 2 Value: 4h — Name: DWORD_COUNT_n **[Default]** — Description: Excludes DWord (0,1)
1	31:4	**Reserved** Format: MBZ Surface Base Address is NOT used for codec H/W. This field is reserved for 3D surface state compatibility. VDEnc pipeline gets this address from VDENC_PIPE_BUF_ADDR_STATE for different buffers.
	3:0	**Reserved**

VDENC_REF_SURFACE_STATE

		Format: MBZ
2..5	127:0	**Dwords 2..5** Format: VDENC_Surface_State_Fields Four DWords that define the bit fields used by the three surface state commands. **Programming Notes** The height and width fields in surface state commands are not used by HW.

VDENC_SRC_SURFACE_STATE

VDENC_SRC_SURFACE_STATE

Source: VideoCS

Length Bias: 2

This command specifies the uncompressed original input picture to be encoded.
The actual base address is defined in the VDENC_PIPE_BUF_ADDR_STATE. 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 in WLV+ VDEnc:

- 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.

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, DMV and streamin/out, a linear buffer is employed. For row stores, the H/W is designed to guarantee legal memory accesses (read and write). For the remaining cases, indirect object base address, indirect object address upper bound, object data start address (offset) and object data length are used to fully specified their corresponding buffer. This mechanism is chosen over the pixel surface type because of their variable record sizes.
All row store surfaces are linear surface. Their addresses are programmed in VDEnc_Pipe_Buf_Base_State.

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
	26:23	**Opcode** Default Value: 1h VDENC_PIPE Format: OpCode
	22:21	**SubOpA** Default Value: 0h Format: OpCode
	20:16	**SubOpB** Default Value: 1h VDENC_SRC_SURFACE_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ

VDENC_SRC_SURFACE_STATE

	11:0	**DWord Length** Format: =n Total Length - 2 **Value** / **Name** / **Description** 4h / DWORD_COUNT_n **[Default]** / Excludes DWord (0,1)
1	31:4	**Reserved** Format: MBZ Surface Base Address is NOT used for codec H/W. This field is reserved for 3D surface state compatibility. VDEnc pipeline gets this address from VDENC_PIPE_BUF_ADDR_STATE for different buffers.
	3:0	**Reserved** Format: MBZ
2..5	127:0	**Dwords 2..5** Format: VDENC_Surface_State_Fields Four DWords that define the bit fields used by the three surface state commands. **Programming Notes** The Width field programmed in DW 2 bit 17:4 should match frame width in pixel, ignoring the comment in Width field

VEB_DI_IECP

VEB_DI_IECP

Source: VideoEnhancementCS

Length Bias: 2

The VEB_DI_IECP command causes the VEBOX to start processing the frames specified by VEB_SURFACE_STATE using the parameters specified by VEB_DI_STATE and VEB_IECP_STATE. The processing can start and end at any 64 pixel column in the frame. If **Starting X** and **Ending X** are used to split the frame into sections, it should not be split into more than 4 sections. Each VEB_DI_IECP command should be preceded by a VEB_STATE command and the input/output VEB_SURFACE_STATE commands.

Programming Notes
When DI is enabled, only the Current Frame skin scores are outputted to the Skin Score Output surface.
When Scalar is enabled only Current frame input and output surface pointers are valid in this command. LACE/ACE histogram pointers are also used if LACE/ACE is enabled. StartX and EndX is not supported with Scalar.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:24	**Opcode** Default Value: 4h VEBOX Format: OpCode
	23:21	**SubOpA** Default Value: 0h VEB_DI_IECP Format: OpCode
	20:16	**SubOpB** Default Value: 3h VEB_DI_IECP Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Total Length - 2 Excludes DWords 0, 1

VEB_DI_IECP

		Value / **Name** 14h / **[Default]**
1	31	**Reserved** Format: MBZ
	30	**Reserved** Format: MBZ
	29:16	**Starting X** Offset from the beginning of the frame to start processing. Must be a multiple of 64 to guarantee that it starts on a column boundary.
	15:14	**Reserved** Format: MBZ
	13:0	**Ending X** Offset from the beginning of the frame to stop processing. Must be a multiple of 64 or equal to the Surface Width to guarantee that it ends on a column boundary. **Programming Notes** **Restriction:**Ending_X - Starting_X must be > = 64.
2	31:12	**Current Frame Input Address** Format: GraphicsAddress[31:12] Specifies bits 31:12 of the 4K byte aligned frame buffer address for reading current frame as input to either the DN/DI or IECP stage.
	11	**Reserved** Format: MBZ
	10:0	**Current Frame Surface Control Bits** Format: VEB_DI_IECP_COMMAND_SURFACE_CONTROL_BITS Please refer to the appropriate project table for Surface Control Bits below.
3	31:30	**Current Frame Input Surface Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. **Value** / **Name** 0 / Highest priority 1 / Second highest priority 2 / Third highest priority 3 / Lowest priority

VEB_DI_IECP

	29:16	**Reserved** Format: MBZ
	15:0	**Current Frame Input Address High** Format: GraphicsAddress[47:32] Specifies bits 47:32 of the address
4	31:12	**Previous Frame Input Address** Format: GraphicsAddress[31:12] Specifies bits 31:12 of the 4K byte aligned frame buffer address for reading the denoised previous frame as input to the DN/DI stage. This field is ignored if **DN Enable**, **DI Enable**, and **Hot Pixel Filtering Enable** are set to 0 (disable).
	11	**Reserved** Format: MBZ
	10:0	**Previous Frame Surface Control Bits** Format: VEB_DI_IECP_COMMAND_SURFACE_CONTROL_BITS
5	31:30	**Previous Frame Input Surface Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.

Value	Name
0	Highest priority
1	Second highest priority
2	Third highest priority
3	Lowest priority

	29:16	**Reserved** Format: MBZ
	15:0	**Previous Frame Input Address High** Format: GraphicsAddress[47:32] Specifies bits 47:32 of the address
6	31:12	**STMM Input Address** Format: GraphicsAddress[31:12] Specifies bits 31:12 of the 4K byte aligned frame buffer address for reading the STMM / denoise history. This field is ignored if **DN Enable**, **DI Enable**, and **Hot Pixel Filtering Enable** are set to 0 (disable).

Programming Notes
This field is ignored if **Disable Temporal Denoise Filter** is set to 1.

VEB_DI_IECP

	11	**Reserved** Format: MBZ
	10:0	**STMM Input Surface Control Bits** Format: VEB_DI_IECP_COMMAND_SURFACE_CONTROL_BITS
7	31:30	**STMM Input Surface Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. **Value** / **Name** 0 – Highest priority 1 – Second highest priority 2 – Third highest priority 3 – Lowest priority
	29:16	**Reserved** Format: MBZ
	15:0	**STMM Input Address High** Format: GraphicsAddress[47:32] Specifies bits 47:32 of the address
8	31:12	**STMM Output Address** Format: GraphicsAddress[31:12] Specifies bits 31:12 of the 4K byte aligned frame buffer address for writing the STMM / denoise history. This field is ignored if **DN Enable**, **DI Enable**, and **Hot Pixel Filtering Enable** are set to 0 (disable). **Programming Notes** This field is ignored if **Disable Temporal Denoise Filter** is set to 1.
	11	**Reserved** Format: MBZ
	10:0	**STMM Output Surface Control Bits** Format: VEB_DI_IECP_COMMAND_SURFACE_CONTROL_BITS
9	31:30	**STMM Output Surface Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. **Value** / **Name** 0 – Highest priority 1 – Second highest priority

		2 — Third highest priority 3 — Lowest priority
	29:16	**Reserved** Format: MBZ
	15:0	**STMM Output Address High** Format: GraphicsAddress[47:32] Specifies bits 47:32 of the address
10	31:12	**Denoised Current Frame Output Address** Format: GraphicsAddress[31:12] Specifies bits 31:12 of the 4K byte aligned frame buffer address for writing the current frame after the DN stage. This field is ignored if both **DN Enable** and **Hot Pixel Filtering Enable** are set to 0 (disable).
	11	**Reserved** Format: MBZ
	10:0	**Denoised Current Output Surface Control Bits** Format: VEB_DI_IECP_COMMAND_SURFACE_CONTROL_BITS Please refer to the appropriate project table for Surface Control Bits below.
11	31:30	**Denoised Current Output Surface Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. 0 — Highest priority 1 — Second highest priority 2 — Third highest priority 3 — Lowest priority
	29:16	**Reserved** Format: MBZ
	15:0	**Denoised Current Frame Output Address High** Format: GraphicsAddress[47:32] Specifies bits 47:32 of the address
12	31:12	**Current Frame Output Address** Format: GraphicsAddress[31:12] Specifies bits 31:12 of the 4K byte aligned frame buffer address for writing the current frame

output. The output is from DN/DI if IECP is disabled, or from IECP if enabled.

11 Reserved

Format:	MBZ

10:0 Current Frame Output Surface Control Bits

Format:	VEB_DI_IECP_COMMAND_SURFACE_CONTROL_BITS

Please refer to the appropriate project table for Surface Control Bits below.

13 31:30 Current Frame Output Surface Arbitration Priority Control

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

Value	Name
0	Highest priority
1	Second highest priority
2	Third highest priority
3	Lowest priority

29:16 Reserved

Format:	MBZ

15:0 Current Frame Output Address High

Format:	GraphicsAddress[47:32]

Specifies bits 47:32 of the address

14 31:12 Previous Frame Output Address

Format:	GraphicsAddress[31:12]

Specifies bits 31:12 of the 4K byte aligned frame buffer address for writing the previous frame output. This field is ignored if **DI Enable** is set to 0 (disable).

11 Reserved

Format:	MBZ

10:0 Previous Frame Output Surface Control Bits

Format:	VEB_DI_IECP_COMMAND_SURFACE_CONTROL_BITS

Please refer to the appropriate project table for Surface Control Bits below.

15 31:30 Previous Frame Output Surface Arbitration Priority Control

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

VEB_DI_IECP

		Value / **Name** 0 — Highest priority 1 — Second highest priority 2 — Third highest priority 3 — Lowest priority
	29:16	**Reserved** Format: MBZ
	15:0	**Previous Frame Output Address High** Format: GraphicsAddress[47:32] Specifies bits 47:32 of the address
16	31:12	**Statistics Output Address** Format: GraphicsAddress[31:12] Specifies bits 31:12 of the 4K byte aligned frame buffer address for writing block level FMD and DN statistics as well as the frame level ACE histogram and FMD frame level statistics.
	11	**Reserved** Format: MBZ
	10:0	**Statistics Output Surface Control Bits** Format: VEB_DI_IECP_COMMAND_SURFACE_CONTROL_BITS Please refer to the appropriate project table for Surface Control Bits below.
17	31:30	**Statistics Output Surface Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. **Value** / **Name** 0 — Highest priority 1 — Second highest priority 2 — Third highest priority 3 — Lowest priority
	29:16	**Reserved** Format: MBZ
	15:0	**Statistics Output Address High** Format: GraphicsAddress[47:32] Specifies bits 47:32 of the address

VEB_DI_IECP

18 — 31:12 — Alpha/Vignette Correction Address

Format:	GraphicsAddress[31:12]

Specifies bits 31:12 of the 4K byte aligned frame buffer address for reading the Alpha surface or vignette correction.

11 — Reserved

Format:	MBZ

10:0 — Alpha/Vignette Control Bits

Format:	VEB_DI_IECP_COMMAND_SURFACE_CONTROL_BITS

Please refer to the appropriate project table for Surface Control Bits below.

19 — 31:30 — Alpha/Vignette Correction Surface Arbitration Priority Control

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

Value	Name
0	Highest priority
1	Second highest priority
2	Third highest priority
3	Lowest priority

29:16 — Reserved

Format:	MBZ

15:0 — Alpha/Vignette Correction Address High

Format:	GraphicsAddress[47:32]

Specifies bits 47:32 of the address

20 — 31:12 — LACE/ACE/RGB Histogram Output Address

Format:	GraphicsAddress[31:12]

Specifies bits 31:12 of the 4K byte aligned address for writing the LACE histograms, or ACE histogram, and/or RGB histogram.

11 — Reserved

Format:	MBZ

10:0 — LACE/ACE/RGB Histogram Control Bits

Format:	VEB_DI_IECP_COMMAND_SURFACE_CONTROL_BITS

		Please refer to the Surface Control Bits Table below.
21	31:30	**LACE/ACE/RGB Histogram Surface Arbitration Priority Control** Format: U2 This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. 0 – Highest priority 1 – Second highest priority 2 – Third highest priority 3 – Lowest priority
	29:16	**Reserved** Format: MBZ
	15:0	**LACE/ACE/RGB Histogram Output Address High** Format: GraphicsAddress[47:32] Bits 47:32 of address.
22	31:12	**Skin Score Output Address** Format: GraphicsAddress[31:12] Specifies bits 31:12 of the 4K byte aligned address for writing the Skin score output in case enabled.
	11	**Reserved** Format: MBZ
	10:0	**Skin Score Output Control Bits** Format: VEB_DI_IECP_COMMAND_SURFACE_CONTROL_BITS Please refer to the Surface Control Bits Table below.
23	31:30	**Skin Score Output Surface Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. Value – Name

		0 Highest priority 1 Second highest priority 2 Third highest priority 3 Lowest priority
	29:16	**Reserved** Format: MBZ
	15:0	**Skin Score Output Address High** Format: GraphicsAddress[47:32] Bits 47:32 of address
24	31:0	**Reserved** Format: MBZ

VEBOX_STATE

VEBOX_STATE

Source: VideoEnhancementCS

Length Bias: 2

Description

This command controls the internal functions of the VEBOX. This command has a set of indirect state buffers:

- DN/DI state
- IECP general state
- IECP Gamut Expansion/Compression state
- IECP Gamut Vertex Table state
- Capture Pipe state

Adds the LACE LUT Table as an indirect state buffer.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:24	**Command OpCode** Default Value: 4h VEBOX Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 2h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Total Length - 2

VEBOX_STATE

		Value / **Name** / **Description**: 11h / / (Excludes DWords 0, 1)
1	31:25	**State Surface Control Bits** All Indirect state buffers use state surface control bits, only exception being 3D LUT state buffer for which the state surface control bits are tied to 0. See definition under "VEB_DI_IECP_COMMAND_SURFACE_CONTROL_BITS" Bits[6:0] is only used.
	24:23	**Reserved** Format: MBZ
	22	**Gamut Expansion Position** If Gamut Expansion is enabled, it can be configured either in front or backend of the IECP pipe using this bit. 0b: Gamut Expansion at the Backend of IECP pipe 1b: Gamut Expansion at the Front of IECP pipe
	21	**Forward Gamma Correction Enable** Format: Enable **Programming Notes** Single Pipe IECP Enable must also be set if this is enabled. When enabled the forward gamma will always be in front of the IECP pipe. In case disabled it will be always configured as Gamut expansion. Gamut Expansion, HDR and Forward Gamma Correction are mutually exclusive.
	20	**Scalar Mode** When Scalar Mode is enabled, all other VEBOX functions must be disabled (DN/DI/DM/IECP/Chroma upsampling).
	19	**Single Pipe Enable** **Description** Indicates that the Capture Pipe features that only exist in a single pipe can be enabled. This bit must be set if any of the following features are enabled: Demosaic Denoise with one of

the RGBA input formats IECP only mode with Forward Gamma Correction enabled with RGB input formats (All other modes are not supported in single pipe)

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

Programming Notes
Note that the pixel throughput is 1/2 when this mode is selected. The **Global IECP Enable** must also be set.

18 **Disable Temporal Denoise Filter**

If set this bit will force the denoise filter to only use the spatial filter. This will eliminate the read of the previous denoise surface and STMM/Denoise History surface and the write of the current denoised surface and STMM/Denoise History surface.

Programming Notes
The Global IECP Enable or **Demosaic Enable** must be set along with this bit. This bit must be set if the input to Denoise is RGB. This bit must not be set if the Deinterlacer is enabled. This bit must be clear if both **DN Enable**=0 and **Hot Pixel Filtering Enable**=0 This bit must be set if **Hot Pixel Filtering Enable**=1 and both DN and DI are disabled

17 **Disable Encoder Statistics**

If set this bit will disable writing the per block Encoder statistics. The memory format is not changed, so the area set aside for these statistics will still be there.

16 **LACE Correction Enable**

This bit enables the correction of the image according to the local ACE LUT tables. This is independent from the enable for the collection of LACE histograms.

Programming Notes
LACE correction is only enabled if both this bit and the **Global IECP Enable** are set. The **ACE Enable** bit should also be set if this bit is set, since ACE correction can be used for part of the luma range instead of LACE.

15:14 **Reserved**

Format:	MBZ

13 **Hot Pixel Filtering Enable**
Enables hot pixel detection/filtering.

12 **Alpha Plane Enable**

Enables the reading of an independent Alpha plane. Mutually exclusive with Vignette Enable. If **Alpha from State Select** is set it overrides this bit.

Programming Notes
IECP must also be enabled and output format must have alpha if this bit is enabled. Should be 0 if Alpha from State Select is 1.

11 **Vignette Enable**

Enables Vignette Correction surface read and correction in IECP. Mutually exclusive with Alpha Plane Enable.

Programming Notes
Demosaic must also be enabled if this bit is enabled.

10 **Demosaic Enable**

The Demosaic will be used, and White balance statistics will be gathered. The Capture Pipe State Table will be read. This bit is mutually exclusive with **DI Enable.**

Programming Notes
IECP must also be enabled if this bit is enabled.

9:8 **DI Output Frames**

Indicates which frames to output in DI mode.

Value	Name
00b	Output Both Frames
01b	Output Previous Frame Only
10b	Output Current Frame Only

Programming Notes
Field is ignored if DI Enable = 0. If Previous Frame Only or Current Frame Only are selected, then the **LACE Single Histogram Set** must not try to collect a histogram from the disabled frame.
Field must be programmed to 10 (Output Current Frame Only) for DI First Frame.

7:6 **Reserved**

Format:	MBZ

5 **DN/DI First Frame**

Format:	Enable

Indicates that this is the first frame of the stream, so previous clean is not available.

Value	Name
0	Not first field; previous clean surface state is valid
1	First field; previous clean surface state is invalid

Programming Notes

		If both DN and DI are disabled, this bit must be 0.
	4	**DI Enable** Format: Enable Deinterlacer is bypassed if this is disabled: the output is the same as the input (same as a 2:2 cadence). FMD and STMM are not calculated and the values in the response message are 0. **Value** / **Name** 0 – Do not calculate DI 1 – Calculate DI
	3	**DN Enable** Format: Enable Denoise is bypassed if this is low - BNE is still calculated and output, but the denoised fields are not. VDI does not read in the denoised previous frame but uses the pointer for the original previous frame. **Value** / **Name** 0 – Do not denoise frame 1 – Denoise frame **Programming Notes** If DN and/or Hotpixel are the only functions enabled then the only output is the Denoised Output which is the same surface format as the input. To get a format conversion with DN only, enable the Global IECP bit, but disable all the individual functions. The IECP output uses the output surface format. If DN is used with RGB then the **Global IECP** Enable must also be
	2	**Global IECP Enable** Indicates if any of the IECP features is enabled. If this is disabled then no state will be read from any of the state pointers. If set then the IECP state will be read.
	1	**Color Gamut Compression Enable** Indicates if the Gamut Compression feature is enabled. If set then the Gamut State will be read. VEB_VERTEXTABLE_STATE is only needed if this bit is set.
	0	**Color Gamut Expansion Enable** **Description** Indicates if the Gamut Expansion feature is enabled. If set then the Gamut State will be read. This can be enabled only if Single pipe enable is disabled.
2	31:12	**DN/DI State Pointer Low** Format: GraphicAddress[31:12] **Description**

		Bits 31:12 of the starting address of the DN/DI State buffer. This points to a buffer containing the 10 Dwords of the DN/DI state. When Scalar mode is enabled this pointer is used for Scalar state table.
	11:0	**Reserved** Format: MBZ
3	31:16	**Reserved** Format: MBZ
	15:0	**DN/DI State Pointer High** Format: GraphicAddress[47:32] **Description** Bits 47:32 of the starting address of the DN/DI State Buffer. When Scalar mode is enabled this pointer is used for Scalar state table.
4	31:12	**IECP State Pointer Low** Format: GraphicAddress[31:12] Bits 31:12 of the starting address of the IECP State buffer. This points to a buffer containing the 64 Dwords of IECP state.
	11:0	**Reserved** Format: MBZ
5	31:16	**Reserved** Format: MBZ
	15:0	**IECP State Pointer High** Format: GraphicAddress[47:32] Bits 47:32 of the starting address of the IECP State Buffer Table.
6	31:12	**Gamut/HDR State Pointer Low** Format: GraphicAddress[31:12] Bits 31:12 of the starting address of the State Buffer. If Gamut Expansion is enabled, this points to a buffer containing the Gamut Expansion Gamma Correction state. If HDR is enabled, this points to a buffer containing the HDR state.
	11:0	**Reserved** Format: MBZ
7	31:16	**Reserved** Format: MBZ
	15:0	**Gamut/HDR State Pointer High**

VEBOX_STATE

		Format: GraphicAddress[47:32] Bits 47:32 of the starting address of the Gamut/HDR State Buffer.
8	31:12	**Vertex Table State Pointer Low** Format: GraphicAddress[31:12] Bits 31:12 of the starting address of the Vertex Table. This points to a buffer containing the 512 Dwords of the Gamut Compression Vertex Table.
	11:0	**Reserved** Format: MBZ
9	31:16	**Reserved** Format: MBZ
	15:0	**Vertex Table State Pointer High** Format: GraphicAddress[47:32] Bits 47:32 of the starting address of the Vertex State Buffer.
10	31:12	**Capture Pipe State Pointer Low** Format: GraphicAddress[31:12] Bits 31:12 of the starting address of the Capture Pipe State Table. This points to a buffer containing the X Dwords of the Capture Pipe State.
	11:0	**Reserved** Format: MBZ
11	31:16	**Reserved** Format: MBZ
	15:0	**Capture Pipe State Pointer High** Format: GraphicAddress[47:32] Bits 47:32 of the starting address of the Capture Pipe State Table.
12	31:12	**LACE LUT Table State Pointer Low** Format: GraphicAddress[31:12] Bits [31:12] of the starting address of the LACE Look-up Tables.
	11:0	**Reserved** Format: MBZ
13	31:30	**Arbitration Priority Control - For LACE LUT**

		Format: U2 This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. **Value** / **Name** 0 – Highest priority 1 – Second highest priority 2 – Third highest priority 3 – Lowest priority
	29:16	**Reserved** Format: MBZ
	15:0	**LACE LUT Table State Pointer High** Format: GraphicAddress[47:32] Bits [47:32] of the starting address of the LACE Look-up Tables.
14..15	63:12	**Gamma Correction Values Address** Format: **GraphicsAddress63-12** Specifies the 4K byte aligned address reading the Gamma Correction Values in case enabled.
	11:0	**Reserved** Format: MBZ
16	31:12	**3D LUT State Pointer Low** Format: GraphicAddress[31:12] Bits [31:12] of the starting address of the 3D LUT.
	11:0	**Reserved** Format: MBZ
17	31:30	**Arbitration Priority Control - For 3D LUT** Format: U2 This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. **Value** / **Name**

		0 Highest Priority 1 Second highest priority 2 Third highest priority 3 Lowest priority
	29	**Reserved**
	21:16	**3D LUT MOCS table** These are surface control bits for VEBOX 3DLUT data requests to GAV>
	15:0	**3D LUT State Pointer High** Format: GraphicAddress[47:32] Bits [47:32] of the starting address of the 3D LUT.
18	31	**3D LUT Enable** Default Value: 0 Format: Enable **3D LUT is required only if this is enabled.**
	30:29	**3D LUT Size** Format: U2

Value	Name
00b	33x33x33
01b	17x17x17
10b	65x65x65

	28:26	**Reserved** Format: MBZ
	25:23	**Reserved** Format: MBZ
	22:14	**Reserved**

	Format:	MBZ

13:12 Frame statistics ID

Format:	U2

This field specifies the Statistics Surface ID number to the VEBOX to writeout the frame statistics.

11 Bypass Chroma Downsampling

Format:	U1

When enabled will drop chroma samples at odd position and not use the co-sited offsets.

Value	Name
0	**[Default]**

10 Bypass Chroma Upsampling

Format:	U1

When enabled will replicate chroma samples at odd position and not use the co-sited offsets.

Value	Name
0	**[Default]**

9:7 Chroma Downsampling Co-Sited Vertical Offset

Format:	U3

Value	Name
0	**[Default]**
[0,2]	Valid Range

6:5 Chroma Downsampling Co-Sited Horizontal Offset

Format:	U2

Value	Name
0	**[Default]**
[0,2]	Valid Range

4:2 Chroma Upsampling Co-Sited Vertical Offset

Format:	U3

VEBOX_STATE

		Value / **Name** 0 / **[Default]** [0,4] / Valid Range
	1:0	**Chroma Upsampling Co-Sited Horizontal Offset** Format: U2 **Value** / **Name** 0 / **[Default]** [0,1] / Valid Range

VEBOX_SURFACE_STATE

VEBOX_SURFACE_STATE	
Source:	VideoEnhancementCS
Length Bias:	2

Description
The input and output data containers accessed are called "surfaces". Surface state is sent to VEBOX via an inline state command rather than using binding tables. SURFACE_STATE contains the parameters defining each surface to be accessed, including its size, format, and offsets to its subsurfaces. The surface's base address is in the execution command. Despite having multiple input and output surfaces, we limit the number of surface states to one for input surfaces and one for output surfaces. The other surfaces are derived from the input/output surface states.
The Current Frame Input surface uses the Input SURFACE_STATE
The Previous Denoised Input surface uses the Input SURFACE_STATE. (For 16-bit Bayer pattern inputs this will be 16-bit.)
The Current Denoised Output surface uses the Input SURFACE_STATE. (For 16-bit Bayer pattern inputs this will be 16-bit.)
The STMM/Noise History Input surface uses the Input SURFACE_STATE with Tile-Y and Width/Height a multiple of 4.
The STMM/Noise History Output surface uses the Input SURFACE_STATE with Tile-Y and Width/Height a multiple of 4.
The Current Deinterlaced/IECP Frame Output surface uses the Output SURFACE_STATE.
The Previous Deinterlaced/IECP Frame Output surface uses the Output SURFACE_STATE.
The FMD per block output / per Frame Output surface uses the Linear SURFACE_STATE (see note below).
The Alpha surface uses the Linear A8 SURFACE_STATE with Width/Height equal to Input Surface. Pitch is width rounded to next 64.
The Skin Score surface uses the Output SURFACE_STATE.
The STMM height is the same as the Input Surface height except when the input **Surface Format** is Bayer Pattern and the **Bayer Pattern Offset** is 10 or 11, in which case the height is the input height + 4. For Bayer pattern inputs when the **Bayer Pattern Offset** is 10 or 11, the Current Denoised Output/Previous Denoised Input will also have a height which is the input height + 4. For Bayer pattern inputs only the Current Denoised Output/Previous Denoised Input are in Tile-Y.
The linear surface for FMD statistics is linear (not tiled). The height of the per block statistics is (Input Height +3)/4 - the Input Surface height in pixels is rounded up to the next even 4 and divided by 4. The width of the per block section in bytes is equal to the width of the Input Surface in pixels rounded up to the next 16 bytes. The pitch of the per block section in bytes is equal to the width of the Input Surface in pixels rounded up to the next 64 bytes.
The STMM surfaces must be identical to the Input surface except for the tiling mode must be Tile-Y and the pitch is specified in DW7. The pitch for the Current Denoised Output/Previous Denoised Input is specified in

VEBOX_SURFACE_STATE

DW7. The width and height must be a multiple of 4 rounded up from the input height.

The Vignette Correction surface uses the Linear 16-bit SURFACE_STATE with :
Width=(Ceil(Image Width / 4) +1) * 4
Height= Ceil(Image Height / 4) +1
Pitch in bytes is (vignette width *2) rounded to the next 64

Programming Notes

VEBOX may write to memory between the surface width and the surface pitch for output surfaces.

VEBOX can support a frame level X/Y offset which allows processing of 2 side-by-side frames for certain 3D video formats.

The X/Y Offset for Frame state applies only to the Current Frame Input and the Current Deinterlaced/IECP Frame Output and Previous Deinterlaced/IECP Frame Output. The statistics surfaces, the denoise feedback surfaces and the alpha/vignette surfaces have no X/Y offsets.

For 8bit Alpha input, when converted to 16bit output, the 8 bit alpha value is replicated to both the upper and lower 8 bits to form the 16 bit alpha value.

Skin Score Output Surface uses the same tiling format as the Output surface.

The input and Output surface for scalar is defined by this command. Derived pitch is not used with the scalar command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Media Command Pipeline** Default Value: 2h Media Format: OpCode
	26:24	**Media Command OpCode** Default Value: 4h VEBOX Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h VEBOX Format: OpCode
	20:16	**SubOpcode B** Default Value: 0h VEBOX Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Total Length - 2

VEBOX_SURFACE_STATE

		Value / **Name** / **Description** 7h / DWORD_COUNT_n **[Default]** / (Excludes DWords 0, 1)
1	31:1	**Reserved** Format: MBZ
	0	**Surface Identification** Specifies which set of surfaces this command refers to: **Value** / **Name** 1 / Output surface (all except the Denoised Current output surface) 0 / Input surface and Denoised Current Output Surface
2	31:18	**Height** Format: U14 This field specifies the height of the surface in units of pixels. For PLANAR surface formats, this field indicates the height of the Y (luma) plane. **Value** / **Name** / **Description** / **Exists If** [15, 16383] / / representing heights [16,16384] / [15, 8191] / / / //Scalar Enabled - For Input surface only [63, 2047] / / / //Scalar + SFC Enabled - For Input surface only **Programming Notes** **Height** (field value + 1) must be a multiple of 2 for PLANAR_420 surfaces. **Height** (field value +1) must be a multiple of 2 when the deinterlace function is enabled (field mode) or when the denoise function is enabled with **Progressive DN** = 0. It must be a multiple of 4 when interleaved deinterlace/denoise and PLANAR_420 are both being used. **VEBOX** supports a minimum height of 16. **Height** (field value + 1) must be a multiple of 2 for Bayer surfaces.
	17:4	**Width** Format: U14 This field specifies the width of the surface in units of pixels. For PLANAR surface formats, this field indicates the width of the Y (luma) plane. **Value** / **Name** / **Description** / **Exists If** [63,16383] / / representing widths [64,16384] / [63,8191] / / / //Scalar Enabled - For Input surface only

[63,2047]			//Scalar and SFC Enabled - For Input Surface only		

Programming Notes

The Width specified by this field multiplied by the pixel size in bytes must be less than or equal to the surface pitch (specified in bytes via the **Surface Pitch** field). **Width** (field value + 1) must be a multiple of 2 for PLANAR_420, PLANAR_422, and all YCRCB_* surfaces, and must be a multiple of 4 for PLANAR_411 surfaces. **VEBOX** supports a minimum width of 64

3:0 **Reserved**

Format:	MBZ

3 31:27 **Surface Format**

Format:	U5

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.

Value	Name	Description
0	YCRCB_NORMAL	
1	YCRCB_SWAPUVY	
2	YCRCB_SWAPUV	
3	YCRCB_SWAPY	
4	PLANAR_420_8	NV12 with Interleave Chroma set
5	PACKED_444A_8	
6	PACKED_422_16	
7	R10G10B10A2_UNORM / R10G10B10A2_UNORM_SRGB	
8	R8G8B8A8_UNORM / R8G8B8A8_UNORM_SRGB	
9	PACKED_444_16	
10	PLANAR_422_16	
11	Y8_UNORM	
12	PLANAR_420_16	
13	R16G16B16A16	
14	Bayer pattern	
15	Y16_UNORM	

26:25 **Bayer Pattern Offset**

Specifies the starting pixel offset for the Bayer pattern used for Capture Pipe.

Value	Name
00b	Pixel at X=0, Y=0 is Blue
01b	Pixel at X=0, Y=0 is Red
10b	Pixel at X=0, Y=0 is Green, Pixel at X=1, Y=0 is Red
11b	Pixel at X=0, Y=0 is Green, Pixel at X=1, Y=0 is Blue

24 **Bayer Pattern Format**

Specifies the format of the Bayer Pattern:

Value	Name
0b	8-bit input at a 8-bit stride
1b	16-bit input at a 16-bit stride

23:22 **Bayer Input Alignment**

Format:	U2

Value	Name
00b	MSB aligned data **[Default]**
01b	10bit LSB aligned data
10b	12bit LSB aligned data
11b	14bit LSB aligned data

Programming Notes
Valid only Bayer Pattern Format is 16bit input

21 **Reserved**

Format:	MBZ

20 **Interleave Chroma**

Format:	Enable

This field indicates that the chroma fields are interleaved in a single plane rather than stored as two separate planes. This field is only used for PLANAR surface formats.

19:3 **Surface Pitch**

Format:	U17 pitch in (Bytes - 1)

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

Value	Name	Description

[63, 131071]	For other linear surfaces	[64B, 128KB]
[511, 131071]	For X-tiled surface	[512B, 128KB] = [1tile, 256 tiles]
[127, 131071]	For Y-tiled surfaces	[128B,128KB] = [1 tile, 1024 tiles]

Programming Notes
For tiled surfaces, the pitch must be a multiple of the tile width. For linear surfaces, the pitch must be a multiple of 64. If Half Pitch for Chroma is set, this field must be a multiple of two tile widths for tiled surfaces, or a multiple of 2 bytes for linear surfaces.

2 **Half Pitch for Chroma**

Format:	Enable

This field indicates that the chroma plane(s) will use a pitch equal to half the value specified in the **Surface Pitch** field. This field is only used for PLANAR surface formats.

Programming Notes
Must be programmed to Zero always as this field is not used

1 **Tiled Surface**

Format:	Boolean

This field specifies whether the surface is tiled.

Value	Name	Description
1	True	Tiled
0	False	Linear

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

0 **Tile Walk**

Format:	3D_TileWalk

This field specifies the type of memory tiling (XMajor or YMajor) employed to tile this surface. See *Memory Interface Functions* for details on memory tiling and restrictions. This field is ignored when the surface is linear.

Value	Name
0	TILEWALK_XMAJOR
1	TILEWALK_YMAJOR

Programming Notes

		The corresponding cache(s) must be invalidated before a previously accessed surface is accessed again with an altered state of this bit.
4	31:29	**Reserved** Format: MBZ
	28:16	**X Offset for U** Format: U13 Pixel Offset This field must be zero for the VEBOX surface formats
	15	**Reserved** Format: MBZ
	14:0	**Y Offset for U** Format: U15 Row Offset This field specifies the vertical offset in rows from the start (origin) or the Luma (Y) plane to the start (origin) of the U(Cb) plane or the interleaved UV plane if **Interleave Chroma** is enabled. This field is only used for PLANAR surface formats. **Programming Notes** This field must indicate an even number (bit 0 = 0). This field must be evenly divisible by 4 for Tile-Y surfaces (so the offset points to the start of a cache line) For Planar formats, if the surface is in YS or YF tile modes, the Y Offset for U should be an integral multiple of the Tile height of the Luma plane
5	31:29	**Reserved** Format: MBZ
	28:16	**X Offset for V** Format: U13 Pixel Offset This field must be zero for the VEBOX surface formats.
	15	**Reserved** Format: MBZ
	14:0	**Y Offset for V** Format: U15 Row Offset This field specifies the vertical offset in rows from the start (origin) of the Luma(Y) plan**e** to the start (origin) of the V(Cr) plane. This field is only used for PLANAR surface formats with **Interleave Chroma** disabled. **Programming Notes** This field must indicate an even number (bit 0 = 0). This field must be evenly divisible by 4 for Tile-Y surfaces (so the offset points to the start of a cache line). For Planar formats, if the surface is in YS or YF tile modes, the Y Offset for V should be an integral multiple of the Tile height of the Luma plane
6	31	**Reserved**

		Format: MBZ
	30:16	**X Offset for Frame** Format: U15 Pixel Offset This is an offset in X from the Surface Base Address in pixels for all planes using this surface. For U/V planes this is added to the X Offset for U/V. After converting to bytes this must be an integer multiple of cache lines in the tiling mode. This specifies the edge of the frame, so adjacent pixels needed for Denoise/Deinterlace/Demosaic are replicated or mirrored. **Programming Notes** If **Y Offset for Frame** >0 the X Offset must be 0. If memory compression is enabled then this must be an even number of cache lines.
	15	**Reserved** Format: MBZ
	14:0	**Y Offset for Frame** Format: U15 Pixel Offset This is an offset in Y from the Surface Base Address in pixels for all planes using this surface. For U/V planes this is added to the Y Offset for U/V. After converting to bytes this must be an integer multiple of cache lines in the tiling mode. This specifies the edge of the frame, so adjacent pixels needed for Denoise/Deinterlace/Demosaic are replicated or mirrored. **Programming Notes** If **X Offset for Frame** >0 the Y Offset must be 0. For Planar formats, if the surface is in YS or YF tile modes, the Y Offset for Frame should be an integral multiple of the Tile height.
7	31:17	**Reserved** Format: MBZ
	16:0	**Derived Surface Pitch** Format: U17 pitch in (Bytes - 1) This field specifies the surface pitch in (#Bytes - 1) for the derived surfaces: STMM/Denoise statistic surface is described when the **Surface Identification** bit is 0 (Input Surface). The (Current Denoise Output)/(Previous Denoise Input) surfaces are described when the bit is 1 (Output Surface).

Value	Name	Description	Exists If
[63, 131071]		[64B, 128KB]	[Tiled Surface] == 0
[511,		[512B, 128KB] = [1tile, 256	([Tiled Surface] == 1) AND ([Tile Walk]

		131071]		tiles]	== 0)
		[127, 131071]		[128B,128KB] = [1 tile, 1024 tiles]	([Tiled Surface] == 1) AND ([Tile Walk] == 1)

Programming Notes
In DN Only mode, the pitch for the (Current Denoise Output)/(Previous Denoise Input) and the Surface Pitch must be programed the same.
The pitch must be a multiple of the tile width.

8 | 31:17 | **Reserved**

Format:	MBZ

16:0 | **Surface Pitch for Skin Score Output Surfaces**

Format:	U17 pitch in (Bytes - 1)

This field specifies the surface pitch in (#Bytes - 1) for the Skin Score Output surface if enabled; This is present only in the output surface format and reserved for Input surface format. The height and width are the same as in the Output surface mentioned above.

Value	Name	Description	Exists If
[63, 131071]		[64B, 128KB]	[Tiled Surface] == 0
[511, 131071]		[512B, 128KB] = [1tile, 256 tiles]	([Tiled Surface] == 1) AND ([Tile Walk] == 0)
[127, 131071]		[128B,128KB] = [1 tile, 1024 tiles]	([Tiled Surface] == 1) AND ([Tile Walk] == 1)

Programming Notes
The pitch must be a multiple of the tile width.

VEBOX_TILING_CONVERT

VEBOX_TILING_CONVERT

Source: VideoEnhancementCS

Length Bias: 2

This command takes the input surface and writes directly to the output surface at high speed. The surface format and width/height of the input and output must be the same, only the tiling mode and pitch can change.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:24	**Command OpCode** Default Value: 4h VEBOX Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 1h Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Total Length - 2 Value: 3h; Name: ; Description: (Excludes DWords 0, 1)
1..2	63:12	**Input Address** Format: GraphicsAddress63-12 Specifies bits 47:12 of the 4Kbyte-aligned frame buffer address for reading the current frame.

<table>
<tr><th colspan="3">VEBOX_TILING_CONVERT</th></tr>
<tr><td></td><td>11</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>10:0</td><td>Input Surface Control Bits<table><tr><td>Format:</td><td>VEB_DI_IECP_COMMAND_SURFACE_CONTROL_BITS</td></tr></table></td></tr>
<tr><td>3..4</td><td>63:12</td><td>Output Address<table><tr><td></td><td></td></tr><tr><td>Format:</td><td>GraphicsAddress63-12</td></tr></table>Specifies bits 47:12 of the 4Kbyte-aligned frame buffer address for writing the current frame.</td></tr>
<tr><td></td><td>11</td><td>Reserved<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>10:0</td><td>Output Surface Control Bits<table><tr><td>Format:</td><td>VEB_DI_IECP_COMMAND_SURFACE_CONTROL_BITS</td></tr></table></td></tr>
</table>

Wait for Event MSD

MSD_WAIT_FOR_EVENT - Wait for Event MSD

Source: EuSubFunctionGateway

Length Bias: 1

Send a writeback if Event ID occurred after MonitorEvent.

DWord	Bit	Description
0	31:29	**Reserved** Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Default Value: 0b Format: Enable **Restriction** Must be zero.
	18:3	**Reserved** Format: MBZ
	2:0	**Wait for Event Subfunction** Default Value: 110b Format: OpCode

Wait Notification

wait - Wait Notification

Source: Eulsa

Length Bias: 4

The wait instruction evaluates the value of the notification count register nreg. If nreg is zero, thread execution is suspended and the thread is put in 'wait_for_notification' state. If nreg is not zero (i.e., one or more notifications have been received), nreg is decremented by one and the thread continues executing on the next instruction. If a thread is in the 'wait_for_notification' state, when a notification arrives, the notification count register is incremented by one. As the notification count register becomes nonzero, the thread wakes up to continue execution and at the same time the notification register is decremented by one. If only one notification arrived, the notification register value becomes zero. However, during the above mentioned time period, it is possible that more notifications may arrive, making the notification register nonzero again. When multiple notifications are received, software must use wait instructions to decrement notification count registers for each notification. Notification register n0.0:ud is for thread to thread communication (via the Message Gateway shared function) and n0.1:ud for host to thread communication (through MMIO registers). See the Message Gateway chapter for thread-thread communication.

Format:

```
wait (exec_size) nreg
```

Restriction
src0 and dst must be n0.0, n0.1, or n0.2.
Execution size must be 1 as the notification registers are scalar.
Predication is not allowed.
Two back-to-back wait instructions are not allowed. At minimum, a nop instruction must be inserted between two wait instructions

Syntax
wait (1) n#

Pseudocode
N/A

Predication	Conditional Modifier	Saturation	Source Modifier
N	N	N	N

Src Types	Dst Types
UD	UD

DWord	Bit	Description
0..3	127:64	**RegSource**

wait - Wait Notification

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

While

while - While

Source: Eulsa

Length Bias: 4

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. If SPF is ON, none of the PcIP are updated.

Format:

```
[(pred)] while (exec_size) JIP
```

Restriction

The execution size must be the same for the while instruction and any break and cont instructions of the same code block.

Syntax

```
[(pred)] while (exec_size) imm32
```

Pseudocode

```
Evaluate(WrEn);
          for ( n = 0; n < 32; n++ ) {
             if (WrEn.chan[n] ) {
                PcIP[n] = IP + JIP;
             }
             else {
                PcIP[n] = IP + 1;
             }
          }

          if ( | PMask == 1 ) {  // any enabled channel true
             Jump(IP + JIP);
          }
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

DWord	Bit	Description
0..3	127:96	**JIP** Format: S31

while - While

		Jump Target Offset. The relative offset in bytes if a jump is taken for the instruction.
	95	**Source 0 Address Immediate [9] Sign Bit**
	94:91	**Src1.SrcType** Format: **SrcType**
	90:89	**Src1.RegFile** Format: **RegFile**
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align16') Format: **EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN16**
	88:64	**Source 0** Exists If: (Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1') Format: **EU_INSTRUCTION_OPERAND_SRC_REG_ALIGN1**
	63:32	**Operand Control** Format: **EU_INSTRUCTION_OPERAND_CONTROLS**
	31:0	**Header** Format: **EU_INSTRUCTION_HEADER**

XY_COLOR_BLT

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.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 50h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. Value / Name: 1xb: Write Alpha Channel x1b: Write RGB Channel
	19:12	**Reserved** Format: MBZ
	11	**Tiling Enable** Value / Name / Description: 0b: Tiling Disabled (Linear Blit) 1b: Tiling Enabled: Tile-X or Tile-Y
	10:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 05h Format: =n
1 BR13	31	**Reserved** Format: MBZ

XY_COLOR_BLT

	30	**Clipping Enabled** Value: 0b — Disabled Value: 1b — Enabled
	29:26	**Reserved** Format: MBZ
	25:24	**Color Depth** 00b — 8 Bit Color 01b — 16 Bit Color(565) 10b — 16 Bit Color(1555) 11b — 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 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..5	63:0	**Destination Base Address** Format: GraphicsAddress63-0 This address must be Cache Line (64Byte) aligned for all surface types (linear or tiled).
6 BR16	31:0	**Solid Pattern Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]

XY_FAST_COPY_BLT

XY_FAST_COPY_BLT

Source: BlitterCS

Length Bias: 2

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 42h Format: Opcode
	21:20	**Source Tiling Method** SW is required to flush the HW before changing the polarity of these bits for subsequent blits. Value / Name / Description: 00b – Linear (Tiling Disabled) 01b – Legacy Tile-X 10b – Tile-Y – Choosing between 'Legacy Tile-Y' or the 'New 4K Tile-YF' can be done in DWord 1, Bit[31]. 11b – 64kb Tiling
	19:15	**Reserved** Format: MBZ
	14:13	**Destination Tiling Method** SW is required to flush the HW before changing the polarity of these bits for subsequent blits. Value / Name / Description: 00b – Linear (Tiling Disabled) 01b – Legacy Tile-X 10b – Tile-Y – Choosing between 'Legacy Tile-Y' or the 'New 4K Tile-YF' can be done in DWord 1, Bit[30]. 11b – 64kb Tiling
	12:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 08h Excludes DWORD 0,1

XY_FAST_COPY_BLT

		Format: =n 08h
1 BR13	31	**Tile Y Type for Source** Source being Tile-Y can be selected in DWord 0, Bit[21:20]. **Value** / **Name** 0b – Legacy Tile-Y 1b – New 4k Tile-YF
	30	**Tile Y Type for Destination** Destination being Tile-Y can be selected in DWord 0, Bit[14:13]. **Value** / **Name** 0b – Legacy Tile-Y 1b – New 4k Tile-YF
	29:27	**Reserved** Format: MBZ
	26:24	**Color Depth** **Value** / **Name** 000b – 8 bit color 001b – 16 bit color (565) 011b – 32 bit color 100b – 64 bit color (for 64KB Tiling) 101b – 128 bit color (for 64KB Tiling)
	23:16	**Reserved** Format: MBZ
	15:0	**Destination Pitch** Format: U16 For **Linear surfaces**, the pitch has to be an **OWord (16byte) multiple** for Linear-Linear copies; and **CacheLine aligned** for Tiled-Linear type copies. The value should be specified as a number in *bytes*. For **Tiled surfaces**, the pitch has to be a **multiple of the Tile width** (X direction width of the Tile). This means the pitch value will always be **Cache Line aligned (64byte multiple**) and the number or value mentioned in this field here should be specified as a number in Dwords (4 byte quantity).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** Format: S16 (16-bit signed number) Destination start line (inclusive) for Fast Copy blit.

XY_FAST_COPY_BLT

	15:0	**Destination X1 Coordinate (Left)** Format: S16 (16-bit signed number) Destination start pixel (inclusive) for Fast Copy blit. It should be on an OWord boundary.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** Format: S16 (16-bit signed number) Destination end line (exclusive) for Fast Copy blit.
	15:0	**Destination X2 Coordinate (Right)** Format: S16 (16-bit signed number) Destination end pixel (exclusive) for Fast Copy blit.
4..5	63:0	**Destination Base Address** Format: **GraphicsAddress63-0** This address must be Cache Line (64Byte) aligned for all surface types (linear or tiled).
6 BR26	31:16	**Source Y1 Coordinate (Top)** Format: S16 (16-bit signed number) Source start line (inclusive) for Fast Copy blit.
	15:0	**Source X1 Coordinate (Left)** Source start pixel (inclusive) for Fast Copy blit. It should be on an OWord boundary.
7 BR11	31:16	**Reserved** Format: MBZ
	15:0	**Source Pitch** Format: U16 For **Linear surfaces**, the pitch has to be an **OWord (16byte) multiple** for Linear-Linear copies; and**CacheLine aligned**for Linear-Tiled type copies. The value should be specified as a number in *bytes*. For **Tiled surfaces**, the pitch has to be a **multiple of the Tile width** (X direction width of the Tile). This means the pitch value will always be **Cache Line aligned (64byte multiple**) and the number or value mentioned in this field here should be specified as a number in Dwords (4 byte quantity).
8..9	63:0	**Source Base Address** Format: **GraphicsAddress63-0** This address must be Cache Line (64Byte) aligned for all surface types (linear or tiled).

XY_FULL_BLT

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.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 55h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. Value / Name: 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:16	**Reserved** Format: MBZ
	15	**Src Tiling Enable** Value / Name / Description: 0b Tiling Disabled (Linear Blit)

1b	Tiling Enabled	Tile-X or Tile-Y.

14:12 Pattern Horizontal Seed
Pixel of the scan line to start on corresponding to DST X=0.

11 Dest Tiling Enable

Value	Name	Description
0b	Tiling Disabled (Linear Blit)	
1b	Tiling Enabled	Tile-X or Tile-Y.

10:8 Pattern Vertical Seed
Starting scan line of the 8x8 pattern corresponding to DST Y=0.

7:0 DWord Length

Default Value:	0Ah

1 BR13

31 Reserved

Format:	MBZ

30 Clipping Enabled

Value	Name
0b	Disabled
1b	Enabled

29:26 Reserved

Format:	MBZ

25:24 Color Depth

Value	Name
00b	8 Bit Color
01b	16 Bit Color(565)
10b	16 Bit Color(1555)
11b	32 Bit Color

23:16 Raster Operation

15:0 Destination Pitch in DWords
2's complement For Tiled surfaces (bit_15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 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.

XY_FULL_BLT

4..5	63:0	**Destination Base Address** Format: GraphicsAddress63-0 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address must be CL (64byte) aligned.
6 BR11	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.
	15:0	**Source Pitch (double word aligned and signed) and in DWords** 2's complement. For Tiled Src (bit 15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).
7 BR26	31:16	**Source Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Source X1 Coordinate (Left)** 16 bit signed number.
8..9	63:0	**Source Address** Format: GraphicsAddress63-0 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address must be CL (64byte) aligned.
10..11	63:0	**Pattern Base Address** Format: GraphicsAddress63-0 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address must be CL (64byte) aligned.

XY_FULL_IMMEDIATE_PATTERN_BLT

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.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 74h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:16	**Reserved** Format: MBZ
	15	**Src Tiling Enable** Value / Name / Description

XY_FULL_IMMEDIATE_PATTERN_BLT

0b	Tiling Disabled (Linear)	
1b	Tiling Enabled	Tile-X or Tile-Y.

14:12 Pattern Horizontal Seed
(pixel of the scan line to start on corresponding to DST X=0)

11 Dest Tiling Enable

Value	Name	Description
0b	Tiling Disabled (Linear Blit)	
1b	Tiling Enabled	Tile-X or Tile-Y.

10:8 Pattern Vertical Seed
Starting scan line of the 8x8 pattern corresponding to DST Y=0.

7:0 DWord Length

Format:	=n

n = 08 + DWL
Where DWL indicates size of immediate data in terms of dwords.

Value	Name
[24,72]	Excludes DWORD 0,1

1 BR13

31 Reserved

Format:	MBZ

30 Clipping Enabled

Value	Name
0b	Disabled
1b	Enabled

29:26 Reserved

Format:	MBZ

25:24 Color Depth

Value	Name
00b	8 Bit Color
01b	16 Bit Color(565)
10b	16 Bit Color(1555)
11b	32 Bit Color

23:16 Raster Operation

15:0 Destination Pitch in DWords
2's complement For Tiled surfaces (bit_15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).

2 BR22

31:16 Destination Y1 Coordinate (Top)
16 bit signed number.

15:0 Destination X1 Coordinate (Left)

XY_FULL_IMMEDIATE_PATTERN_BLT		
		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..5	63:0	**Destination Base Address** Format: GraphicsAddress63-0 Base address of the destination surface: X=0, Y=0. When Src Tiling is enabled (Bit_15 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.
6 BR11	31:16	**Reserved** Format: MBZ Should be programmed all 0's for 48bit addressing.
	15:0	**Source Pitch (double word aligned and signed) and in DWords** 2's complement. For Tiled Src (bit 15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).
7 BR26	31:16	**Source Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Source X1 Coordinate (Left)** 16 bit signed number.
8..9	63:0	**Source Address** Format: GraphicsAddress63-0 Base address of the source surface: X=0, Y=0. When Src Tiling is enabled (Bit_15 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.
10..n	31:0	**Immediate Data 0**

XY_FULL_MONO_PATTERN_BLT

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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="4">0
BR00</td><td>31:29</td><td>Client<table><tr><td>Default Value:</td><td>02h 2D Processor</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td>28:22</td><td>Instruction Target(Opcode)<table><tr><td>Default Value:</td><td>57h</td></tr><tr><td>Format:</td><td>Opcode</td></tr></table></td></tr>
<tr><td>21:20</td><td>32bpp Byte Mask
This field is only used for 32bpp.<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>[Default]</td></tr><tr><td>1xb</td><td>Write Alpha Channel</td></tr><tr><td>x1b</td><td>Write RGB Channel</td></tr></table></td></tr>
<tr><td>19:16</td><td>Reserved</td></tr>
</table>

XY_FULL_MONO_PATTERN_BLT

	15	**Src Tiling Enable** Value / Name / Description 0b / Tiling Disabled (Linear Blit) / 1b / Tiling Enabled / Tile-X or Tile-Y.
	14:12	**Pattern Horizontal Seed** (pixel of the scan line to start on corresponding to DST X=0)
	11	**Dest Tiling Enable** Value / Name / Description 0b / Tiling Disabled (Linear Blit) / 1b / Tiling Enabled / Tile-X or Tile-Y.
	10:8	**Pattern Vertical Seed** Starting scan line of the 8x8 pattern corresponding to DST Y=0.
	7:0	**DWord Length** Value / Name 0Ch /
1 BR13	31	**Solid Pattern Select** Value / Name 0 / No Solid Pattern 1 / Solid Pattern
	30	**Clipping Enabled** Value / Name 0b / Disabled 1b / Enabled
	29	**Reserved** Format: / MBZ
	28:27	**Mono Source Transparency Mode** Value / Name 0 / Use Background 1 / Transparency Enabled
	26	**Reserved** Format: / MBZ
	25:24	**Color Depth** Value / Name 00b / 8 Bit Color 01b / 16 Bit Color(565) 10b / 16 Bit Color(1555)

XY_FULL_MONO_PATTERN_BLT

		11b: 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 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..5	63:0	**Destination Base Address** Format: GraphicsAddress63-0 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_15 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.
6 BR11	31:16	**Reserved** Format: MBZ
	15:0	**Source Pitch (double word aligned and signed) and in DWords** 2's complement. For Tiled Src (bit 15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).
7 BR26	31:16	**Source Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Source X1 Coordinate (Left)** 16 bit signed number.
8..9	63:0	**Source Address** Format: GraphicsAddress63-0 Base address of the source surface: X=0, Y=0. When Src Tiling is enabled (Bit_15 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.
10 BR16	31:0	**Pattern Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
11 BR17	31:0	**Pattern Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
12 BR20	31:0	**Pattern Data 0** (least significant DW)
13 BR21	31:0	**Pattern Data 1** (most significant DW)

XY_FULL_MONO_PATTERN_MONO_SRC_BLT

XY_FULL_MONO_PATTERN_MONO_SRC_BLT

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)** Default Value: 58h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:17	**Monochrome source data bit position of the first pixel within a**

		byte per scan line.
	16:15	**Reserved** Format: MBZ
	14:12	**Pattern Horizontal Seed** (pixel of the scan line to start on corresponding to DST X=0)
	11	**Tiling Enable** Value: 0b – Name: Tiling Disabled (Linear Blit) Value: 1b – Name: Tiling Enabled – Description: Tile-X or Tile-Y.
	10:8	**Pattern Vertical Seed** Starting scan line of the 8x8 pattern corresponding to DST Y = 0.
	7:0	**DWord Length** Value: 0Ch
1 BR13	31	**Solid Pattern Select** Value: 0 – Name: No Solid Pattern Value: 1 – Name: Solid Pattern
	30	**Clipping Enabled** Value: 0b – Name: Disabled Value: 1b – Name: Enabled
	29	**Mono Source Transparency Mode** Value: 0 – Name: Use Background Value: 1 – Name: Transparency Enabled
	28	**Mono Pattern Transparency Mode** Value: 0 – Name: Use Background Value: 1 – Name: Transparency Enabled
	27:26	**Reserved** Format: MBZ
	25:24	**Color Depth** Value: 00b – Name: 8 Bit Color Value: 01b – Name: 16 Bit Color(565)

		10b: 16 Bit Color(1555) 11b: 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 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..5 This bitfield contains the base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	**Destination Base Address** Format: GraphicsAddress63-0
6..7 Address corresponds to DST X1, Y1. Note no NPO2 change here. The Mono Source Base Address must always be Cache Line (64byte) aligned.	63:0	**Mono Source Address** Format: GraphicsAddress63-0
8 BR18	31:0	**Source Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
9 BR19	31:0	**Source Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
10 BR16	31:0	**Pattern Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
11 BR17	31:0	**Pattern Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]

XY_FULL_MONO_PATTERN_MONO_SRC_BLT		
12 BR20	31:0	**Pattern Data 0** (least significant DW)
13 BR21	31:0	**Pattern Data 1** (most significant DW)

XY_FULL_MONO_SRC_BLT

XY_FULL_MONO_SRC_BLT

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

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 56h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. Value / Name 00b / **[Default]** 1xb / Write Alpha Channel x1b / Write RGB Channel
	19:17	**Monochrome source data bit position of the first pixel within a byte per scan line.**
	16:15	**Reserved** Format: MBZ
	14:12	**Pattern Horizontal Seed**

XY_FULL_MONO_SRC_BLT

		(pixel of the scan line to start on corresponding to DST X=0)
	11	**Tiling Enable** Value / Name / Description 0b / Tiling Disabled (Linear Blit) / 1b / Tiling Enabled / Tile-X or Tile-Y.
	10:8	**Pattern Vertical Seed** Starting scan line of the 8x8 pattern corresponding to DST Y = 0.
	7:0	**DWord Length** Value / Name 0Ah /
1 BR13	31	**Reserved** Format: / MBZ
	30	**Clipping Enabled** Value / Name 0b / Disabled 1b / Enabled
	29	**Mono Source Transparency Mode** Value / Name 0 / Use Background 1 / Transparency Enabled
	28:26	**Reserved** Format: / MBZ
	25:24	**Color Depth** Value / Name 00b / 8 Bit Color 01b / 16 Bit Color(565) 10b / 16 Bit Color(1555) 11b / 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.

<table>
<tr><th colspan="3">XY_FULL_MONO_SRC_BLT</th></tr>
<tr><td></td><td>15:0</td><td>Destination X1 Coordinate (Left)
16 bit signed number.</td></tr>
<tr><td rowspan="2">3
BR23</td><td>31:16</td><td>Destination Y2 Coordinate (Bottom)
16 bit signed number.</td></tr>
<tr><td>15:0</td><td>Destination X2 Coordinate (Right)
16 bit signed number.</td></tr>
<tr><td>4..5
Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address must be CL (64byte) aligned.</td><td>63:0</td><td>Destination Base Address
<table><tr><td></td><td></td></tr><tr><td>Format:</td><td>GraphicsAddress63-0</td></tr></table></td></tr>
<tr><td>6..7
Address corresponds to DST X1, Y1. Note no NPO2 change here. The Mono Source Base Address must always be Cache Line (64byte) aligned.</td><td>63:0</td><td>Mono Source Address
<table><tr><td></td><td></td></tr><tr><td>Format:</td><td>GraphicsAddress63-0</td></tr></table></td></tr>
<tr><td>8
BR18</td><td>31:0</td><td>Source Background Color
8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]</td></tr>
<tr><td>9
BR19</td><td>31:0</td><td>Source Foreground Color
8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]</td></tr>
<tr><td>10..11
28:06 are implemented. Note no NPO2 change here. The pattern data must be located in linear memory. The programmed Pattern Base Address must always be Cache Line (64byte) aligned.</td><td>63:0</td><td>Pattern Base Address
<table><tr><td></td><td></td></tr><tr><td>Format:</td><td>GraphicsAddress63-0</td></tr></table></td></tr>
</table>

XY_FULL_MONO_SRC_IMMEDIATE_PATTERN_BLT

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.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 75h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. Value / Name: 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:17	**Monochrome source data bit position of the first pixel within a byte per scan line.**
	16:15	**Reserved** Format: MBZ
	14:12	**Pattern Horizontal Seed** (pixel of the scan line to start on corresponding to DST

X=0)

11 Tiling Enable

Value	Name	Description
0b	Tiling Disabled (Linear Blit)	
1b	Tiling Enabled	Tile-X or Tile-Y.

10:8 Pattern Vertical Seed

Starting scan line of the 8x8 pattern corresponding to DST Y=0.

7:0 DWord Length

Format:	=n

n = 08 + DWL

Where DWL indicates immediate data sizes in dwords.

Value	Name
[24,72]	Excludes DWORD 0,1

1 BR13

31 Reserved

Format:	MBZ

30 Clipping Enabled

Value	Name
0b	Disabled
1b	Enabled

29 Mono Source Transparency Mode

Value	Name
0	Use Background
1	Transparency Enabled

28:26 Reserved

Format:	MBZ

25:24 Color Depth

Value	Name
00b	8 Bit Color
01b	16 Bit Color(565)
10b	16 Bit Color(1555)
11b	32 Bit Color

23:16 Raster Operation

15:0 Destination Pitch in DWords

2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).

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..5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	**Destination Base Address** Format: GraphicsAddress63-0
6..7 Address corresponds to DST X1, Y1. Note no NPO2 change here. The Mono Source Base Address must always be Cache Line (64byte) aligned.	63:0	**Mono Source Address** Format: GraphicsAddress63-0
8 BR18	31:0	**Source Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
9 BR19	31:0	**Source Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
10..n	31:0	**Immediate Data**

XY_MONO_PAT_BLT

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.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 52h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:15	**Reserved** Format: MBZ
	14:12	**Pattern Horizontal Seed** Pixel of the scan line to start on corresponding to DST X=0.
	11	**Tiling Enable** 0b Tiling Disabled (Linear Blit) 1b Tiling Enabled — Tile-X or Tile-Y.

XY_MONO_PAT_BLT

	10:8	**Pattern Vertical Seed** Scan line of the 8x8 pattern to start on corresponding to DST Y=0.
	7:0	**DWord Length** Value: 08h Name:
1 BR13	31	**Reserved** Format: MBZ
	30	**Clipping Enabled** 0b: Disabled 1b: Enabled
	29	**Reserved** Format: MBZ
	28	**Mono Pattern Transparency Mode** 0: Use Background 1: Transparency Enabled
	27:26	**Reserved** Format: MBZ
	25:24	**Color Depth** 00b: 8 Bit Color 01b: 16 Bit Color(565) 10b: 16 Bit Color(1555) 11b: 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 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.

XY_MONO_PAT_BLT		
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4..5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	**Destination Base Address** Format: GraphicsAddress63-0
6 BR16	31:0	**Pattern Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
7 BR17	31:0	**Pattern Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
8 BR20	31:0	**Pattern Data 0**
9 BR21	31:0	**Pattern Data 1**

XY_MONO_PAT_FIXED_BLT

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.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 59h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. Value / Name: 00b – **[Default]** 1xb – Write Alpha Channel x1b – Write RGB Channel
	19	**Reserved** Format: MBZ
	18:15	**Fixed Pattern** Value / Name: 0000b – HS_HORIZONTAL 0001b – HS_VERTICAL 0010b – HS_FDIAGONAL

		0011b: HS_BDIAGONAL 0100b: HS_CROSS 0101b: HS_DIAGCROSS 0110b: Reserved 0111b: Reserved 1000b: Screen Door 1001b: SD Wide 1010b: Walking Bit (one) 1011b: Walking Zero 1100b: Reserved 1101b: Reserved 1110b: Reserved 1111b: Reserved
	14:12	**Pattern Horizontal Seed** Pixel of the scan line to start on corresponding to DST X=0.
	11	**Tiling Enable** Value 0b: Tiling Disabled (Linear Blit) Value 1b: Tiling Enabled — Tile-X or Tile-Y.
	10:8	**Pattern Vertical Seed** Scan line of the 8x8 pattern to start on corresponding to DST Y=0.
	7:0	**DWord Length** Format: =n Value 06h: Name
1 BR13	31	**Reserved** Format: MBZ
	30	**Clipping Enabled** Value 0b: Disabled Value 1b: Enabled
	29	**Reserved** Format: MBZ
	28	**Mono Pattern Transparency Mode** Value / Name

		0 — Use Background 1 — Transparency Enabled
	27:26	**Reserved** Format: MBZ
	25:24	**Color Depth** 00b — 8 Bit Color 01b — 16 Bit Color(565) 10b — 16 Bit Color(1555) 11b — 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 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..5	63:0	**Destination Base Address** Format: GraphicsAddress63-0 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.
6 BR16	31:0	**Pattern Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
7 BR17	31:0	**Pattern Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]

XY_MONO_SRC_COPY_BLT

XY_MONO_SRC_COPY_BLT

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.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 54h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:17	**Monochrome source data bit position of the first pixel within a byte per scan line.**
	16:12	**Reserved** Format: MBZ
	11	**Tiling Enable** 0b Tiling Disabled (Linear Blit) 1b Tiling Enabled — Tile-X or Tile-Y.

XY_MONO_SRC_COPY_BLT

	10:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Value: 08h; Name:
1 BR13	31	**Reserved** Format: MBZ
	30	**Clipping Enabled** 0b: Disabled 1b: Enabled
	29	**Mono Source Transparency Mode** 0: Use Background 1: Transparency Enabled
	28:26	**Reserved** Format: MBZ
	25:24	**Color Depth** 00b: 8 Bit Color 01b: 16 Bit Color(565) 10b: 16 Bit Color(1555) 11b: 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 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..5	63:0	**Destination Base Address**

Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.		Format: GraphicsAddress63-0
6..7 Address corresponds to DST X1, Y1. Note no NPO2 change here. The Mono Source Base Address must always be Cache Line (64byte) aligned.	63:0	**Mono Source Address** Format: GraphicsAddress63-0
8 BR18	31:0	**Source Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
9 BR19	31:0	**Source Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]

XY_MONO_SRC_COPY_IMMEDIATE_BLT

XY_MONO_SRC_COPY_IMMEDIATE_BLT

Source: BlitterCS

Length Bias: 2

This instruction allows the Driver to send monochrome data through the instruction stream, eliminating the read latency of the source during command execution.
The IMMEDIATE_BLT data MUST transfer an even number of doublewords and the exact number of quadwords. DWL indicates the total number of Dwords of immediate data.
All non-text monochrome sources are word aligned. At the end of a scan line of monochrome source, all bits until the next word boundary are ignored. The Monochrome source data bit position field [2:0] indicates the bit position within the first byte of the scan line that should be used as the first source pixel which corresponds to the destination X1 coordinate.
The monochrome source transparency mode indicates whether to use the source background color or de-assert the write enables when the bit in the source is 0. When the source bit is 1, then the source foreground color is used in the ROP operation. The ROP value chosen must involve source and no pattern data in the ROP operation.
The monochrome source data supplied corresponds to the Destination X1 and Y1 coordinates.
Negative Stride (= Pitch) is NOT ALLOWED.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 71h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. Value / Name 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:17	**Monochrome source data bit position of the first pixel within a byte per scan line.**
	16:12	**Reserved** Format: MBZ
	11	**Tiling Enable** Value / Name / Description 0b Tiling Disabled (Linear)

		1b: Tiling Enabled — Tile-X or Tile-Y.
	10:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: [22,70] Excludes DWORD 0,1 Format: =n n = 06 + DWL Where DWL indicates the total number of Dwords of immediate data.
1 BR13	31	**Reserved** Format: MBZ
	30	**Clipping Enabled** 0b: Disabled 1b: Enabled
	29	**Mono Source Transparency Mode** 0b: Transparency Enabled 1b: Use Background
	28:26	**Reserved** Format: MBZ
	25:24	**Color Depth** 00b: 8 Bit Color 01b: 16 Bit Color(565) 10b: 16 Bit Color(1555) 11b: 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 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.

XY_MONO_SRC_COPY_IMMEDIATE_BLT		
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4..5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	**Destination Base Address** Format: **GraphicsAddress63-0**
6 BR18	31:0	**Source Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
7 BR19	31:0	**Source Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
8..n	31:0	**Immediate Data**

XY_PAT_BLT

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.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 51h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:15	**Reserved** Format: MBZ
	14:12	**Pattern Horizontal Seed** Pixel of the scan line to start on corresponding to DST X=0.
	11	**Tiling Enable** 0b Tiling Disabled (Linear Blit) 1b Tiling Enabled — Tile-X or Tile-Y.
	10:8	**Pattern Vertical Seed** Scan line of the 8x8 pattern to start on corresponding to DST Y=0.
	7:0	**DWord Length** Default Value: 06h
1	31	**Reserved**

XY_PAT_BLT

BR13		Format: MBZ
	30	**Clipping Enabled** Value: 0b — Name: Disabled Value: 1b — Name: Enabled
	29:26	**Reserved** Format: MBZ
	25:24	**Color Depth** Value: 00b — Name: 8 Bit Color Value: 01b — Name: 16 Bit Color(565) Value: 10b — Name: 16 Bit Color(1555) Value: 11b — Name: 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 up to 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..5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	**Destination Base Address** Format: GraphicsAddress63-0
6..7 28:06 are implemented. Note no NPO2 change here. The pattern data must be located in linear memory. The programmed Pattern Base Address must always be Cache Line (64byte) aligned.	63:0	**Pattern Base Address** Format: GraphicsAddress63-0

XY_PAT_BLT_IMMEDIATE

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.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 72h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:15	**Reserved** Format: MBZ
	14:12	**Pattern Horizontal Seed** Pixel of the scan line to start on corresponding to DST X=0.
	11	**Tiling Enable** 0b Tiling Disabled (Linear Blit) 1b Tiling Enabled — Tile-X or Tile-Y.
	10:8	**Pattern Vertical Seed** Scan line of the 8x8 pattern to start on corresponding to DST

XY_PAT_BLT_IMMEDIATE

		Y=0.
	7:0	**DWord Length** Default Value: [20,68] Excludes DWORD 0,1 Format: =n n = 04 + DWL Where DWL indicates number of immediate data in terms of dwords.
1 BR13	31	**Reserved** Format: MBZ
	30	**Clipping Enabled** 0b: Disabled 1b: Enabled
	29:26	**Reserved** Format: MBZ
	25:24	**Color Depth** 00b: 8 Bit Color 01b: 16 Bit Color(565) 10b: 16 Bit Color(1555) 11b: 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement (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 up to 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..5 Base address of the destination surface:	63:0	**Destination Base Address**

XY_PAT_BLT_IMMEDIATE		
X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.		Format: **GraphicsAddress63-0**
6..n	31:0	**Immediate Data**

XY_PAT_CHROMA_BLT

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.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 76h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:17	**Transparency Range Mode** (chroma-key) - Dst Chroma-key modes ONLY (SRC ILLEGAL)
	16:15	**Reserved** Format: MBZ
	14:12	**Pattern Horizontal Seed** Pixel of the scan line to start on corresponding to DST X=0.
	11	**Tiling Enable** 0b Tiling Disabled (Linear Blit) 1b Tiling Enabled — Tile-X or Tile-Y.
	10:8	**Pattern Vertical Seed** Scan line of the 8x8 pattern to start on corresponding to DST Y=0.
	7:0	**DWord Length**

XY_PAT_CHROMA_BLT

		Default Value: 08h Excludes DWORD 0,1
1 BR13	31	**Reserved** Format: MBZ
	30	**Clipping Enabled** 0b: Disabled 1b: Enabled
	29:26	**Reserved** Format: MBZ
	25:24	**Color Depth** 00b: 8 Bit Color 01b: 16 Bit Color(565) 10b: 16 Bit Color(1555) 11b: 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement (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 up to 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..5	63:0	**Destination Base Address** Format: GraphicsAddress63-0 This address must be Cache Line (64Byte) aligned for all surface types (linear or tiled).
6..7	63:0	**Pattern Base Address** Format: GraphicsAddress63-0 (28:06 are implemented) (Note no NPO2 change here). The pattern data must be located in linear memory. The Pattern Base Address must always be Cache Line (64byte) aligned.

XY_PAT_CHROMA_BLT		
8 BR18	31:0	**Transparency Color Low** (Chroma-key Low = Pixel Greater or Equal)
9 BR19	31:0	**Transparency Color High** (Chroma-key High = Pixel Less or Equal)

XY_PAT_CHROMA_BLT_IMMEDIATE

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.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 77h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:17	**Transparency Range Mode** (chroma-key) - Dst Chroma-key modes ONLY (SRC ILLEGAL)
	16:15	**Reserved** Format: MBZ
	14:12	**Pattern Horizontal Seed** Pixel of the scan line to start on corresponding to DST X=0.
	11	**Tiling Enable** 0b Tiling Disabled (Linear Blit)

XY_PAT_CHROMA_BLT_IMMEDIATE

		1b: Tiling Enabled — Tile-X or Tile-Y.
	10:8	**Pattern Vertical Seed** Scan line of the 8x8 pattern to start on corresponding to DST Y=0.
	7:0	**DWord Length** Format: =n n = 06 + DWL Where DWL indicates the total number of Dwords of immediate data. Value: [22,72] — Name: Excludes DWORD 0,1
1 BR13	31	**Reserved** Format: MBZ
	30	**Clipping Enabled** 0b: Disabled 1b: Enabled
	29:26	**Reserved** Format: MBZ
	25:24	**Color Depth** 00b: 8 Bit Color 01b: 16 Bit Color(565) 10b: 16 Bit Color(1555) 11b: 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 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.

XY_PAT_CHROMA_BLT_IMMEDIATE		
4..5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	**Destination Base Address** Format: **GraphicsAddress63-0**
6 BR18	31:0	**Transparency Color Low** (Chroma-key Low = Pixel Greater or Equal)
7 BR19	31:0	**Transparency Color High** (Chroma-key High = Pixel Less or Equal)
8..n	31:0	**Immediate Data**

XY_PIXEL_BLT

XY_PIXEL_BLT

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

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.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 25h Format: Opcode
	21:15	**Reserved** Format: MBZ
	14:12	**Pattern Horizontal Seed** Pixel of the scan line to start on corresponding to DST X=0.
	11	**Tiling Enable** Value / Name / Description 0b / Tiling Disabled (Linear Blit) / 1b / Tiling Enabled / Tile-X or Tile-Y.
	10:8	**Pattern Vertical Seed** Scan line of the 8x8 pattern to start on corresponding to DST Y=0.
	7:0	**DWord Length** Default Value: 01h
1 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
2 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)**

XY_SCANLINES_BLT		
		16 bit signed number.

XY_SETUP_BLT

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).

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 01h Format: Opcode
	21:20	**32 bpp Byte Mask** Value / Name 1xb: Write Alpha Channel x1b: Write RGB Channel
	19:12	**Reserved** 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: 08h
1 BR01	31	**Reserved** Format: MBZ
	30	**Clipping Enabled**

XY_SETUP_BLT

		Value: 0b — Name: Disabled Value: 1b — Name: Enabled
	29	**Mono Source Transparency Mode** Value: 0b — Name: Use Background Value: 1b — Name: Transparency Enabled
	28:26	**Reserved** Format: MBZ
	25:24	**Color Depth** Value: 00b — Name: 8 Bit Color Value: 01b — Name: 16 Bit Color(565) Value: 10b — Name: 16 Bit Color(1555) Value: 11b — Name: 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 up to 128Kbytes (or 32KDwords).
2 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)
3 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)
4..5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	**Destination Base Address** Format: **GraphicsAddress63-0**
6 BR05	31:0	**Setup Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0] All
7 BR06	31:0	**Setup Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0] (SLB and TB only)

<table>
<tr><th colspan="3">XY_SETUP_BLT</th></tr>
<tr><td>8..9
28:06 are implemented. Note no NPO2 change here. The pattern data must be located in linear memory. The Setup Pattern Base Address for Color Pattern must always be Cache Line (64byte) aligned.</td><td>63:0</td><td>Setup Pattern Base Address for Color Pattern<table><tr><td></td><td></td></tr><tr><td>Format:</td><td>GraphicsAddress63-0</td></tr></table></td></tr>
</table>

XY_SETUP_CLIP_BLT

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: 0b — Name: Tiling Disabled (Linear Blit) Value: 1b — Name: Tiling Enabled (Tile-X or Tile-Y
	10:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 01h
1 BR24	31:16	**ClipRect Y1 Coordinate (Top)** (30:16 = 15 bit positive number)
	15:0	**ClipRect X1 Coordinate (Left)** (14:00 = 15 bit positive number)
2 BR25	31:16	**ClipRect Y2 Coordinate (Bottom)** (30:16 = 15 bit positive number)
	15:0	**ClipRect X2 Coordinate (Right)** (14:00 = 15 bit positive number)

XY_SETUP_MONO_PATTERN_SL_BLT

XY_SETUP_MONO_PATTERN_SL_BLT

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** 1xb: Write Alpha Channel x1b: Write RGB Channel
	19:12	**Reserved** Format: MBZ
	11	**Tiling Enable** 0b: Tiling Disabled (Linear Blit) 1b: Tiling Enabled (Tile-X or Tile-Y
	10:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 08h
1 BR01	31	**Solid Pattern Select** (SLB and Pixel only) 0: No Solid Pattern 1: Solid Pattern
	30	**Clipping Enabled**

XY_SETUP_MONO_PATTERN_SL_BLT

		Value: 0b — Disabled Value: 1b — Enabled
	29	**Reserved** Format: MBZ
	28	**Mono Pattern Transparency Mode** Value: 0b — Use Background Value: 1b — Transparency Enabled
	27:26	**Reserved** Format: MBZ
	25:24	**Color Depth** Value: 00b — 8 Bit Color Value: 01b — 16 Bit Color(565) Value: 10b — 16 Bit Color(1555) Value: 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 up to 128Kbytes (or 32KDwords).
2 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)
3 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)
4..5 This bitfield contains the base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	**Setup Destination Base Address** Format: GraphicsAddress63-0

XY_SETUP_MONO_PATTERN_SL_BLT		
6 BR05	31:0	**Setup Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0] All
7 BR06	31:0	**Setup Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0] (SLB and TB only)
8 BR20	31:0	**DW0 (least significant) for a Monochrome Pattern**
9 BR21	31:0	**DW1 (most significant) for a Monochrome Pattern**

XY_SRC_COPY_BLT

XY_SRC_COPY_BLT

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.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 53h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:16	**Reserved** Format: MBZ
	15	**Src Tiling Enable** 0b Tiling Disabled (Linear) 1b Tiling Enabled — Tile-X or Tile-Y.
	14:12	**Reserved** Format: MBZ
	11	**Dest Tiling Enable**

XY_SRC_COPY_BLT

		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** Format: =n **Value**: 08h — **Name**:
1 BR13	31	**Reserved** Format: MBZ
	30	**Clipping Enabled** 0b — Disabled 1b — Enabled
	29:26	**Reserved** Format: MBZ
	25:24	**Color Depth** 00b — 8 Bit Color 01b — 16 Bit Color(565) 10b — 16 Bit Color(1555) 11b — 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 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.

XY_SRC_COPY_BLT

4..5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	**Destination Base Address** Format: **GraphicsAddress63-0**
6 BR26	31:16	**Source Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Source X1 Coordinate (Left)** 16 bit signed number.
7 BR11	31:16	**Reserved** Format: MBZ
	15:0	**Source Pitch (double word aligned) and in DWords** 2's complement. For Tiled Src (bit 15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).
8..9 Base address of the source surface: X=0, Y=0. When Src Tiling is enabled (Bit_15 enabled), this address must be 4KB-aligned. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	**Source Base Address** Format: GraphicsAddres63-0

XY_SRC_COPY_CHROMA_BLT

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.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 73h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:17	**Transparency Range Mode** (chroma-key)
	16	**Reserved** Format: MBZ
	15	**Src Tiling Enable** 0b Tiling Disabled (Linear) 1b Tiling Enabled — Tile-X or Tile-Y.
	14:12	**Reserved**

XY_SRC_COPY_CHROMA_BLT

		Format: MBZ
	11	**Dest Tiling Enable** 0b: Tiling Disabled (Linear Blit) 1b: Tiling Enabled — Tile-X or Tile-Y.
	10:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Value: 0Ah
1 BR13	31	**Reserved** Format: MBZ
	30	**Clipping Enabled** 0b: Disabled 1b: Enabled
	29:26	**Reserved** Format: MBZ
	25:24	**Color Depth** 00b: 8 Bit Color 01b: 16 Bit Color(565) 10b: 16 Bit Color(1555) 11b: 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 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.

XY_SRC_COPY_CHROMA_BLT

4..5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	**Destination Base Address** Format: GraphicsAddress63-0
6 BR26	31:16	**Source Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Source X1 Coordinate (Left)** 16 bit signed number.
7 BR11	31:16	**Reserved** Format: MBZ
	15:0	**Source Pitch (double word aligned) and in DWords** 2's complement. For Tiled Src (bit 15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).
8..9 Base address of the source surface: X=0, Y=0. When Src Tiling is enabled (Bit_15 enabled), this address must be 4KB-aligned. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	**Source Base Address** Format: GraphicsAddress63-0
10 BR18	31:0	**Transparency Color Low** (Chroma-key Low = Pixel Greater or Equal)
11 BR19	31:0	**Transparency Color High** (Chroma-key High = Pixel Less or Equal)

XY_TEXT_BLT

XY_TEXT_BLT

Source: BlitterCS

Length Bias: 2

All source scan lines and pixels that fall within the ClipRect Y and X coordinates are written. The source address corresponds to Destination X1 and Y1 coordinate.
Text is either bit or byte packed. Bit packed means that the next scan line starts 1 pixel after the end of the current scan line with no bit padding. Byte packed means that the next scan line starts on the first bit of the next byte boundary after the last bit of the current line.
Source expansion color registers are always in the SETUP_BLT.
Negative Stride (= Pitch) is NOT ALLOWED.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 26h Format: Opcode
	21:17	**Reserved** Format: MBZ
	16	**Bit / Byte Packed** Byte packed is for the NT driver. Value: 0 = Bit; 1 = Byte
	15:12	**Reserved** Format: MBZ
	11	**Tiling Enable** 0b: Tiling Disabled (Linear Blit) 1b: Tiling Enabled — Tile-X or Tile-Y.
	10:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 03h

<table>
<tr><th colspan="3">XY_TEXT_BLT</th></tr>
<tr><td rowspan="2">1
BR22</td><td>31:16</td><td>Destination Y1 Coordinate (Top)
16 bit signed number.</td></tr>
<tr><td>15:0</td><td>Destination X1 Coordinate (Left)
16 bit signed number.</td></tr>
<tr><td rowspan="2">2
BR23</td><td>31:16</td><td>Destination Y2 Coordinate (Bottom)
16 bit signed number.</td></tr>
<tr><td>15:0</td><td>Destination X2 Coordinate (Right)
16 bit signed number.</td></tr>
<tr><td>3..4
Address of the first byte on a scan line corresponding to source X1, Y1. Note no NPO2 change here. The source address must always be Cache Line (64byte) aligned.</td><td>63:0</td><td>Source Address
<table><tr><td></td><td></td></tr><tr><td>Format:</td><td>GraphicsAddress63-0</td></tr></table></td></tr>
</table>

XY_TEXT_IMMEDIATE_BLT

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 0: Bit Value 1: Byte
	15:12	**Reserved** Format: MBZ
	11	**Tiling Enable** 0b: Tiling Disabled (Linear Blit) 1b: Tiling Enabled — Tile-X or Tile-Y.
	10:8	**Reserved** Format: MBZ
	7:0	**DWord Length**

XY_TEXT_IMMEDIATE_BLT		
		Default Value: [17,65] Excludes DWORD 0,1 Format: =n n = 01 + DWL Where DWL is the number of immediate data in dwords.
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**