Intel® Open Source HD Graphics

Programmer's Reference Manual

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

Volume 2c: Command Reference: Structures

May 2017, Revision 1.0

Creative Commons License

Notices and Disclaimers

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

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

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

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

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

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

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

Table of Contents

Any Binding Table Index Message Descriptor Control Field

MDC_BTS_SLM_A32 - Any Binding Table Index Message Descriptor Control Field

Source:	BSpec
Size (in bits):	8
Default Value:	0x00000000

DWord	Bit	Description
0	7:0	**Binding Table Index**

Format:	Enumeration

Specifies the surface for the message, which can be Surface State Model, SLM or Stateless.

Value	Name	Description
00h-0EFh	BTS	Index of Binding Table State Surfaces
F0h-0FBh	Reserved	Reserved for future use
0FCh	Reserved	Reserved for future use
0FEh	SLM	Specifies an SLM access
0FFh	A32_A64	Specifies a A32 or A64 Stateless access that is locally coherent (coherent within a thread group)
0FDh	A32_A64_NC	Specifies a A32 or A64 Stateless access that is non-coherent (coherent within a thread).

Restriction
Restriction : When using A32_A64_NC, SW must ensure that 2 threads do not both access the same cache line (64B)

Bit Definition for Interrupt Control Registers - Render

<table>
<tr><th colspan="3">Bit Definition for Interrupt Control Registers - Render</th></tr>
<tr><td colspan="3">Source: RenderCS
Size (in bits): 32
Default Value: 0x00000000</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:16</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table>Reserved for other command streamers - cannot be allocated by main command streamer.</td></tr>
<tr><td></td><td>15:12</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11</td><td><b>Wait on Semaphore</b>
Exec-List Scheduling: Set 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.</td></tr>
<tr><td></td><td>10</td><td><b>L3 Counter Save Interrupt</b></td></tr>
<tr><td></td><td>9</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>8</td><td><b>Context Switch Interrupt</b>
Set when a context switch has just occurred. Execlist Enable bit needs to be set for this interrupt to occur.</td></tr>
<tr><td></td><td>7</td><td><b>Page Fault</b><table><tr><th>Description</th></tr><tr><td>This interrupt is for handling Legacy Page Fault interface for all Command Streamers (BCS, RCS, VCS, VECS). When Fault Repair Mode is enabled, Interrupt mask register value is not looked at to generate interrupt due to page fault. Please refer to vol1c "Page Fault Support" section for more details.</td></tr></table></td></tr>
<tr><td></td><td>6</td><td><b>Timeout Counter Expired</b>
Set when the render pipe timeout counter (0x02190) has reached the timeout threshold value (0x0217c).</td></tr>
<tr><td></td><td>5</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>4</td><td><b>PIPE_CONTROL Notify Interrupt</b>
The Pipe Control packet (Fences) specified in 3D pipeline document may optionally generate an Interrupt. The Store QW associated with a fence is completed ahead of the interrupt.</td></tr>
<tr><td></td><td>3</td><td><b>Render Command Parser Master Error</b>
When this status bit is set, it indicates that the hardware has detected an error. It is set by the device upon an error condition and cleared by a CPU write of a one to the appropriate bit contained in the Error ID register followed by a write of a one to this bit in the IIR. Further</td></tr>
</table>

Bit Definition for Interrupt Control Registers - Render

		information on the source of the error comes from the "Error Status Register" which along with the "Error Mask Register" determine which error conditions will cause the error status bit to be set and the interrupt to occur. **Page Table Error:** Indicates a page table error. **Instruction Parser Error:** The Render Instruction Parser encounters an error while parsing an instruction.
	2	**Reserved** Format: MBZ
	1	**Reserved**
	0	**Render Command Parser User Interrupt** This status bit is set when an MI_USER_INTERRUPT instruction is executed on the Render Command Parser. Note that instruction execution is not halted and proceeds normally. A mechanism such as an MI_STORE_DATA instruction is required to associate a particular meaning to a user interrupt.

Context Descriptor Format

Context Descriptor Format	
Source:	BSpec
Size (in bits):	64
Default Value:	0x00000000, 0x00000000
This is the format of context descriptors which make up submitted execlists.	

DWord	Bit	Description
0..1	63:32	**Context ID** **Description** Context ID is a unique field assigned by GFX driver when a new context is created by which it is identified across all hierarchies of SW and HW. • Context ID is used for semaphore signaling by hardware and software. • Context ID matching is used by hardware to detect Lite Restore. • Context ID is used by hardware for page fault reporting and response with IOMMU. • Context switch reason and the associated Context ID are reported to Context Switch Status Buffer by hardware on a context switch. Context ID[31:0] (bits[63:32] of the context descriptor) are used for comparing during lite restore, semaphore signaling and context specific OA enabling. Context ID which is a 32 bit field is further divided in to three segments described below: • **Bits[63:55] (Bits 31:23 of Context ID)** is referred to as GroupID. GroupId+PASID combination of a context must be a unique identifier for contexts that are active in the system. The definition of active context is listed as: • Any Context that is already submitted to h/w or already running in h/w. • Any Context that hit page faults, was preempted (didn't run to context complete), and is waiting to be resubmitted pending IOMMU "last in group" response. • Any Context that has experienced reset but not all faults are responded to. • **Bit[54] (Bit 22 of Context ID)** – MBZ for SW programming; this bit is used by hardware to distinguish between F&H vs F&S page requests and response messages to and from IOMMU. This bit is used by hardware on receiving page response to properly manage the page fault counters • **Bit[53] (Bit 21 of Context ID)** – MBZ from SW programming, is reserved for future hardware use. • **Bits[52:32] (Bits 20:0 of Context ID)** are for software use-only and must be unique field assigned by GFX driver when a new context is created. Context ID is reported by hardware to OABUFFER along with the performance statistics counters, Context ID is used for filtering the statistics on per context basis.

Context Descriptor Format

31:12 **Logical Ring Context Address (LRCA)**

Format:	GraphicsAddress[31:12]

This field contains the 4 KB-aligned address of the Logical Ring Context associated with this execlist element. LRCA must be always programmed in GGTT memory.

11:9 **Reserved**

Format:	MBZ

8 **Privilege Access**

This field when set indicates PPGTT enabled in legacy context mode.
In advanced context mode this field is reserved and must be zero.

7:6 **Fault Handling**

Source:	CommandStreamer

Value	Name	Description
0h	Fault and Hang	Fault model is not supported and fault occurrence is treated as catastrophic. GAM indicates Fault Error to Command streamer. Fault Error interrupt is reported to scheduler. Command Streamer will not initiate context switch on occurrence of Fault Error.
1h	Reserved	Reserved
2h	Fault and Stream	In this mode of operation faults are allowed on EU memory accesses. Page Walker will directly work with memory page handler to fix the faults on the fly for these surfaces. Command streamer is not aware of the fault service being done by page walker and goes with its normal execution rules for context switch. On completion of flush during context switch CS explicitly requests acknowledge message to Page Walker before proceeding further. Page Walker acknowledges Command Streamer once it is done on a clean boundary. Page Walker asserts Fault Error on occurrence of non recoverable fault or access violations (Command Streamer access, VFunit access, etc) to Command Streamer; this is the same as Fault and Hang behavior.
Others	Reserved	Reserved

Programming Notes
When execlist mode is set to "Legacy Context mode" Fault Handling mode must be set to "Fault and Hang." For proper programming for Page Fault modes, refer to memory interface section of the Bspec for the corresponding generation.

5 **Reserved**

Format:	MBZ

Context Descriptor Format

4:3 **Addressing Mode & Legacy Context**

Format:	U2

Legacy context set indicates GPU is operating in legacy context mode of operation and doesn't support any SVM features. Legacy context reset indicates GPU is operating in advanced context mode of operation and support SVM features. Based on the Context mode set Addressing mode is interpreted appropriately. The table below summarizes the combinations supported. GFX engine always uses 32b virtual addressing mode when translated using GGTT irrespective of below options.

Value	Name	Description
00b	Advanced Context with no A/D support	GPU is enabled for advanced context mode and supports SVM features. GPU DOESN'T support Access and Dirty bit management in page tables. GPU supports 64b(48bit canonical) PPGTT graphics virtual addressing. PDP0_DESCRIPTOR contains the PASID (process address space identifier) and other PDP Descriptors are ignored.
01b	Legacy Context with no 64 bit VA support	GPU is enabled for legacy context mode of operation and DOESN'T support any SVM features. GPU supports 32b PPGTT graphics virtual addressing. PDP*_DESCRIPTOR contains the base address to 4GB of memory space supported.
10b	Advanced Context with A/D support	GPU is enabled for advanced context mode and supports SVM features. GPU DOES support Access and Dirty bit management in page tables. GPU supports 64b (48bit canonical) PPGTT graphics virtual addressing. PDP0_DESCRIPTOR contains the PASID (process address space identifier) and other PDP Descriptors are ignored.
11b	Legacy Context with 64 bit VA support	GPU is enabled for legacy context mode of operation and DOESN'T support any SVM features. GPU supports 64b (48bit canonical) PPGTT graphics virtual addressing and PDP0_DESCRIPTOR contains the base address to PML4 and other PDP Descriptors are ignored.

2 **Force Restore**

Setting this bit will force a context restore operation when switching to this context even if the LRCA in the CCID register (normally the LRCA of the last context from the prior execlist) matches this one.

Note that it is legal (and likely desirable) for the **Render Context Restore Inhibit** bit (part of the CTXT_SR_CTL register) in the context image being restored to also be set. The "ring" context is being forced to be restored from a newly initialized context despite a possible LRCA match. However, the render context for such a newly initialized context will likely be uninitialized and so should not be restored.

Workaround	Source
Workaround Force Restore bit must be always be set on all context submissions.	BlitterCS, VideoCS, VideoEnhancementCS

Context Descriptor Format		
	1	**Force PD Restore** Setting this bit will cause the on-chip page directory to be reloaded from the PD image in memory even on an LRCA match. No other operations of context restore will occur on an LRCA match, however. Software should set this bit if it has updated a context's page directory and wants the context to begin using the new page directory without having to switch away from it (to another context) and back again. Setting this bit will have no effect if **Force Restore** is also set; a complete context restore (including the PD) will be performed.
	0	**Valid** Set if this register holds a valid context descriptor. SW should set this bit in the Element registers that it has set up to contain valid context descriptors. Any execlist elements that are not used in a submitted execlist must have this bit clear.

HW Generated BINDING_TABLE_STATE

HW Generated BINDING_TABLE_STATE

Source:	BSpec
Size (in bits):	16
Default Value:	0x00000000

Description
The binding table binds surfaces to logical resource indices used by shaders and other compute engine kernels. The HW generated Binding_Table_State have different format than the SW generated Binding_Table_State. The HW generated Binding_Table_State is stored as an array of 256 elements, each of which contains one word as defined here. The start of each element is spaced one word apart. The first element of the binding table is aligned to a 64-byte boundary. Binding table indexes beyond 256 will automatically be mapped to entry 0 by the HW, w/ the exception of any messages which support the special indexes 255 and 254.

DWord	Bit	Description
0	15:0	**Surface State Pointer** Format: SurfaceStateOffset[21:6]

Image_State_Cost

Image_State_Cost

Source:	VideoCS
Size (in bits):	64
Default Value:	0x00000000, 0x00000000

DWord	Bit	Description
0	31:24	**MV 3 Cost** Format: U4U4 **Programming Notes** U4U4 format explanation
	23:16	**MV 2 Cost** Format: U4U4
	15:8	**MV 1 Cost** Format: U4U4
	7:0	**MV 0 Cost** Format: U4U4
1	31:24	**MV 7 Cost** Format: U4U4
	23:16	**MV 6 Cost** Format: U4U4
	15:8	**MV 5 Cost** Format: U4U4
	7:0	**MV 4 Cost** Format: U4U4

INTERFACE_DESCRIPTOR_DATA

<table>
<tr><th colspan="3">INTERFACE_DESCRIPTOR_DATA</th></tr>
<tr><td colspan="3">Source: RenderCS
Size (in bits): 256
Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:6</td><td><b>Kernel Start Pointer</b>
<table><tr><td>Format:</td><td>InstructionBaseOffset[31:6]Kernel</td></tr></table>
Specifies the 64-byte aligned address offset of the first instruction in the kernel. This pointer is relative to the <b>Instruction Base Address.</b></td></tr>
<tr><td></td><td>5:0</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>1</td><td>31:16</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>15:0</td><td><b>Kernel Start Pointer High</b>
<table><tr><td>Format:</td><td>InstructionBaseOffset[47:32]Kernel</td></tr></table>
This field specifies the high 16 bits of starting address of the Kernel Pointer.</td></tr>
<tr><td>2</td><td>31:20</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>19</td><td><b>Denorm Mode</b>
This field specifies how Float denormalized numbers are handles in the dispatched thread.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Ftz</td><td>Float denorms will be flushed to zero when appearing as inputs; denorms will never come out of instructions. Double precision float and half precision float numbers are not flushed to zero.</td></tr>
<tr><td>1h</td><td>SetByKernel</td><td>Denorms will be handled in by kernel.</td></tr>
</table></td></tr>
<tr><td></td><td>18</td><td><b>Single Program Flow</b>
Specifies whether the kernel program has a single program flow (SIMDnxm with m = 1) or multiple program flows (SIMDnxm with m > 1).
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0h</td><td>Multiple</td></tr>
<tr><td>1h</td><td>Single</td></tr>
</table></td></tr>
<tr><td></td><td>17</td><td><b>Thread Priority</b>
Specifies the priority of the thread for dispatch.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0h</td><td>Normal Priority</td></tr>
</table></td></tr>
</table>

<table>
<tr><th colspan="3">INTERFACE_DESCRIPTOR_DATA</th></tr>
<tr><td></td><td></td><td><table><tr><td>1h</td><td>High Priority</td></tr></table></td></tr>
<tr><td></td><td>16</td><td><b>Floating Point Mode</b>
Specifies the floating point mode used by the dispatched thread.<table><tr><th>Value</th><th>Name</th></tr><tr><td>0h</td><td>IEEE-754</td></tr><tr><td>1h</td><td>Alternate</td></tr></table></td></tr>
<tr><td></td><td>15:14</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>13</td><td><b>Illegal Opcode Exception Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>This bit gets loaded into EU CR0.1[12] (note the bit # difference). See <i>Exceptions and ISA Execution Environment</i>.</td></tr>
<tr><td></td><td>12</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11</td><td><b>Mask Stack Exception Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>This bit gets loaded into EU CR0.1[11]. See <i>Exceptions and ISA Execution Environment</i>.</td></tr>
<tr><td></td><td>10:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7</td><td><b>Software Exception Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>This bit gets loaded into EU CR0.1[13] (note the bit # difference). See <i>Exceptions and ISA Execution Environment</i>.</td></tr>
<tr><td></td><td>6:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>3</td><td>31:5</td><td><b>Sampler State Pointer</b><table><tr><td>Format:</td><td>DynamicStateOffset[31:5]SAMPLER_STATE</td></tr></table>Specifies the 32-byte aligned address offset of the sampler state table. This pointer is relative to the <b>Dynamic State Base Address.</b>
<i>This field is ignored for child threads.</i></td></tr>
</table>

INTERFACE_DESCRIPTOR_DATA

<table>
<tr><td></td><td>4:2</td><td><b>Sampler Count</b>
<table><tr><td>Format:</td><td>U3</td></tr></table>
Specifies how many samplers (in multiples of 4) the kernel uses. Used only for prefetching the associated sampler state entries.
<i>This field is ignored for child threads.</i>
<i>If this field is not zero, sampler state is prefetched for the first instance of a root thread upon the startup of the media pipeline.</i>
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>[0,4]</td><td></td></tr>
<tr><td>0h</td><td>No samplers used</td></tr>
<tr><td>1h</td><td>Between 1 and 4 samplers used</td></tr>
<tr><td>2h</td><td>Between 5 and 8 samplers used</td></tr>
<tr><td>3h</td><td>Between 9 and 12 samplers used</td></tr>
<tr><td>4h</td><td>Between 13 and 16 samplers used</td></tr>
</table></td></tr>
<tr><td></td><td>1:0</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>4</td><td>31:16</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>15:5</td><td><b>Binding Table Pointer</b>
<table><tr><td>Format:</td><td>SurfaceStateOffset[15:5]BINDING_TABLE_STATE*256</td></tr></table>
<table>
<tr><th>Description</th></tr>
<tr><td>Specifies the 32-byte aligned address of the binding table. This pointer is relative to the <b>Surface State Base Address</b>. <i>This field is ignored for child threads.</i></td></tr>
</table></td></tr>
<tr><td></td><td>4:0</td><td><b>Binding Table Entry Count</b>
<table><tr><td>Format:</td><td>U5</td></tr></table>
Specifies how many binding table entries the kernel uses. Used only for prefetching of the binding table entries and associated surface state.
<i>This field is ignored for child threads.</i>
<i>If this field is not zero, binding table and surface state are prefetched for the first instance of a root thread upon the startup of the media pipeline.</i>
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>[0,31]</td><td></td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>The maximum number of prefetched binding table entries is limited to 31. For kernels using a large number of binding table entries, it may be wise to set this field to zero to avoid prefetching too many entries and thrashing the state cache.</td></tr>
</table></td></tr>
</table>

INTERFACE_DESCRIPTOR_DATA

5 | 31:16 | **Constant/Indirect URB Entry Read Length**

Format:	U16

Specifies the amount of URB data read and passed in the thread payload for the Constant or Indirect URB entry, in 8-DW register increments. A value 0 means that no Constant or Indirect URB Entry will be loaded. The Constant URB Entry Read Offset field will then be ignored. In GPGPU mode this describes how much data is delivered in a single dispatch. Multiple dispatches in a thread group will deliver constant data offset by this value. The total amount of constant data is (Constant URB Read Length * Number of Threads in GPGPU Thread Group + Cross-Thread Constant Data Read Length).

If **Cross-Thread Constant Data Read Length** for Indirect is greater than 0, then this field must also be greater than 0. The allowed combinations are:

Constant/Indirect URB Entry Read Length **Entry Read Length**	Cross-Thread Constant **Data Read Length**	Notes
=0	=0	No Payload
>0	=0	Per-thread payload only
>0	>0	Both kinds of payload
=0	>0	Only for CURBE payloads

Value	Name
[0,63]	

15:0 | **Constant URB Entry Read Offset**

Format:	U16

Specifies the offset (in 8-DW units) at which Constant URB data is to be read from the URB before being included in the thread payload.

Value	Name	Description
[0,1983]		Indicating [0,1983] 256-bit register increments. ROB has 64KB of storage; 2048 entries. However, lowest 64 entries are reserved for VFE/TS to store interface descriptor data. Hence, (URB Entry Read Offset + Read Length) shall not exceed 1984.

6 | 31:24 | **Reserved**

Format:	MBZ

23:22 | **Rounding Mode**

Format:	U2

Value	Name	Description
00b	RTNE **[Default]**	Round to Nearest Even

INTERFACE_DESCRIPTOR_DATA

01b	RU	Round toward +Infinity
10b	RD	Round toward -Infinity
11b	RTZ	Round toward Zero

21 **Barrier Enable**

Format:	Enable

This field specifies whether the thread group requires a barrier. If not, it can be dispatched without allocating one.

20:16 **Shared Local Memory Size**

Format:	U5

This field indicates how much Shared Local Memory the thread group requires. The amount is specified in 4k blocks, but only powers of 2 are allowed: 0, 4k, 8k, 16k, 32k and 64k per half-slice.
Uses a different encoding to allow encodings for the new 1k and 2k SLM sizes.

Value	Name	Description
0	Encodes 0K	No SLM used
1	Encodes 1K	
2	Encodes 2K	
3	Encodes 4K	
4	Encodes 8K	
5	Encodes 16K	
6	Encodes 32K	
7	Encodes 64K	

15 **Global Barrier Enable**

Format:	Enable

Description
This field when set indicates that the thread group associated with this barrier is allowed to cross sub-slices with a performance penalty. When this field is clear, the thread group is dispatched to a single sub-slice or pool. Note that SLM should not be used with a Global Barrier since SLM is always forced to a single sub-slice or pool. The Barrier Enable bit must be set to enable the barrier, since this bit only specifies the type of barrier.
Restriction : Global barriers cannot be used. Must be zero.

14:13 **Reserved**

Format:	MBZ

12:10 **Reserved**

Format:	MBZ

<table>
<tr><th colspan="3">INTERFACE_DESCRIPTOR_DATA</th></tr>
<tr><td></td><td>9:0</td><td><b>Number of Threads in GPGPU Thread Group</b>
<table><tr><td>Format:</td><td>U10</td></tr></table>
Specifies the number of threads that are in this thread group.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>[1,54]</td><td></td><td>The minimum value is 1, while the maximum value is the number of threads in a pool or subslice for local barriers. See vol1b Configurations for the number of threads per subslice for different products. The number of threads per pool is defined by the MEDIA_POOL_STATE command.</td></tr>
</table>
<table>
<tr><th>Restriction</th></tr>
<tr><td>Restriction : The maximum value for global barriers is limited by the number of threads in the system, or by 511, whichever is lower. This field should not be set to 0 even if the barrier is disabled, since an accurate value is needed for proper pre-emption.</td></tr>
</table></td></tr>
<tr><td>7</td><td>31:8</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>Cross-Thread Constant Data Read Length</b>
<table><tr><td>Format:</td><td>U8</td></tr></table>
Specifies the amount of constant data in CURBE in 8-DW register increments which will be sent to every thread in the thread group in addition to the per thread ids specified by <b>Constant URB Entry Read Length</b>.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>[0,127]</td><td></td></tr>
</table></td></tr>
</table>

MEDIA_SURFACE_STATE

<table>
<tr><th colspan="3">MEDIA_SURFACE_STATE</th></tr>
<tr><td colspan="3">
<table>
<tr><td>Source:</td><td>BSpec</td></tr>
<tr><td>Exists If:</td><td>//([MessageType] == 'Deinterlace') OR ([MessageType] == 'Sample_8x8')</td></tr>
<tr><td>Size (in bits):</td><td>256</td></tr>
<tr><td>Default Value:</td><td>0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000</td></tr>
</table>
</td></tr>
<tr><td colspan="3">This is the SURFACE_STATE used by only deinterlace, sample_8x8, and VME messages.</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:30</td><td><b>Rotation</b>
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>00b</td><td>No Rotation or 0 Degree</td></tr>
<tr><td>01b</td><td>90 Degree Rotation</td></tr>
<tr><td>10b</td><td>180 Degree Rotation</td></tr>
<tr><td>11b</td><td>270 Degree Rotation</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>Rotation is only supported only with AVS function messages and not with HDC direct write and 16x8 AVS messages.</td></tr>
</table>
</td></tr>
<tr><td></td><td>29:27</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table>
</td></tr>
<tr><td></td><td>26:20</td><td><b>X Offset</b>
<table>
<tr><td>Exists If:</td><td>//[Surface Format] is one of Planar Formats</td></tr>
<tr><td>Format:</td><td>PixelOffset[8:2]</td></tr>
</table>
This field specifies the horizontal offset in pixels from the <b>Surface Base Address</b> to the start (origin) of the surface.
This field effectively loosens the alignment restrictions on the origin of tiled surfaces. Previously, tiled surface origin was (by definition) located at the base address, and thus needed to satisfy the 4KB base address alignment restriction. Now the origin can be specified at a finer (4-wide x 4-high pixel) resolution.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>[0,508]</td><td></td><td>In multiples of 4 (low 2 bits missing)</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>For linear surfaces and Packed Formats, this field must be zero.</td></tr>
<tr><td>For <b>Surface Format</b> with 8 bits per element, this field must be a multiple of 16.</td></tr>
<tr><td>For <b>Surface Format</b> with 16 bits per element, this field must be a multiple of 8.</td></tr>
</table>
</td></tr>
</table>

MEDIA_SURFACE_STATE

26:16 **Reserved**

Exists If:	//[Surface Format] is not one of Planar Formats
Format:	MBZ

19:16 **Y Offset**

Exists If:	//[Surface Format] is one of Planar Formats
Format:	RowOffset[5:2]

This field specifies the vertical offset in rows from the **Surface Base Address** to the start of the surface. (See additional description in the **X Offset** field)

Value	Name	Description
[0,28]		In multiples of 4 (low two bits missing)

Programming Notes
For linear surfaces and Packed Formats, this field must be zero.

15:12 **Reserved**

Format:	MBZ

11:0 **Reserved**

Format:	MBZ

1

31:18 **Height**

Format:	U14-1

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
[0,16383]		representing heights [1,16384]	[Surface Type] != FM_STRBUF_*

Programming Notes
Height (field value + 1) must be a multiple of 2 for PLANAR_420 surfaces.If Vertical Line Stride is 1, this field indicates the height of the field, not the height of the frame.

17:4 **Width**

Format:	U14-1

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
[0,16383]		representing widths [1,16384]	[Surface Type] != FM_STRBUF_*

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

MEDIA_SURFACE_STATE

YCRCB_* and Y16_UNORM surfaces, and must be a multiple of 4 for PLANAR_411 and Y8_UNORM_VA surfaces.

- For deinterlace messages, the Width (field value + 1) must be a multiple of 8.

- For Y8_UNORM_VA format width should be in multiple of 4, for Y16_UNORM_VA format width should be in multiple of 2, for Y1_UNORM format width should be in multiple of 32
- When Address Control = Mirror, the total width should be in multiple of 4bytes.

Width (field value + 1) must be a multiple of 2 for PLANAR_420_16

3:2 **Picture Structure**

Specifies the encoding of the current picture.

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

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

Default Value:	0
Format:	U0.2

Description
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
This offset has been increased from 2 bits to 3 bits to support U1.2 format, and the MSB bit is added as Pixel Offset V Direction MSB in DWord 2. Valid values for the combined field range from 0 to 4.

2 31:27 **Surface Format**

Description
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.
Note: Y8_UNORM_VA, Y16_UNORM and Y16_SNORM are used for all functions of sample_8x8 except AVS where rest of the formats are not used. These two formats are packed as 32bits in L1 though the individual pixels are either 8bpp or 16bpp respectively.

Value	Name	Description

MEDIA_SURFACE_STATE

0	YCRCB_NORMAL	
1	YCRCB_SWAPUVY	
2	YCRCB_SWAPUV	
3	YCRCB_SWAPY	
4	PLANAR_420_8	
5	Y8_UNORM_VA	Sample_8x8 only except AVS
6	Y16_SNORM	Sample_8x8 only except AVS
7	Y16_UNORM_VA	Sample_8x8 only except AVS
8	R10G10B10A2_UNORM	Sample_8x8 only
9	R8G8B8A8_UNORM	Sample_8x8 AVS only
10	R8B8_UNORM (CrCb)	Sample_8x8 AVS only
11	R8_UNORM (Cr/Cb)	Sample_8x8 AVS only
12	Y8_UNORM	Sample_8x8 AVS only
13	A8Y8U8V8_UNORM	Sample_8x8 AVS only
14	B8G8R8A8_UNORM	Sample_8x8 AVS only
15	R16G16B16A16	Sample_8x8 AVS only
16	Y1_UNORM	Sample_8x8 only for boolean surfaces (1bit/pixel)
17	Y32_UNORM	For Integral Image (32bpp)
18	PLANAR_422_8	Sample_8x8 AVS only
Others	Reserved	

26

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.

25

Cr(V)/Cb(U) Pixel Offset U Direction

Default Value:	0
Format:	U0.1

Description
Specifies the distance to the U/V values with respect to the even numbered Y channels in the U direction

Programming Notes
This field must be zero for all formats except PLANAR_420_8, PLANAR_422_8, YCRCB_NORMAL, YCRCB_SWAPUVY, YCRCB_SWAPUV, YCRCB_SWAPY.

MEDIA_SURFACE_STATE

24 **Cr(V)/Cb(U) Pixel Offset V Direction MSB**

Default Value:	0
Format:	U1

Description
Specifies the distance to the U/V values with respect to the even numbered Y channels in the V direction

Programming Notes
This field must be zero for all formats except PLANAR_420_8.
This offset has been increased from 2 bits to 3 bits as U1.2 format and this bit is used in conjunction with the bits in the Cr(V)/Cb(U) Pixel Offset V Direction field in DWord 1, which contain the rest of the bits for offset V-direction. Valid values for the combined field range from 0 to 4.

23 **Memory Compression Mode**

Distinguishes Vertical from Horizontal compression.

Value	Name	Description
0	Horizontal Compression Mode **[Default]**	
1	Vertical Compression Mode	

22 **Memory Compression Enable**

Format:	Enable

This surface may contain compressed or compressible pixels. Memory compression will be attempted for writes to this surface. Reads from this surface will check for compressed data.

Programming Notes
The compression control must have 0 value for non-tileY modes.
Please refer to vol1a Memory Data Formats chapter -- section Media Memory Compression for more details, including format restrictions.

21 **Address Control**

Value	Name	Description
0	CLAMP	Clamp
1	MIRROR	Mirror

20:3 **Surface Pitch**

Format:	U18-1 pitch in Bytes

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

Value	Name	Description
[0,262143]		For other linear surfaces: representing [1B, 256KB]
[511, 262143]		For X-tiled surface: representing [512B, 256KB] = [1 tile, 512 tiles]

MEDIA_SURFACE_STATE

[127, 262143]		For Y-tiled surfaces: representing [128B, 256KB] = [1 tile, 2048 tiles]

Programming Notes
For tiled surfaces, the pitch must be a multiple of the tile widthIf 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.The Surface Pitches of current picture and reference picture should be declared as the identical type in VDI mode with identical Height, Width and Format.
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

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 Zero as this field is not used.

1:0

Tile Mode

Format:	U2 Enumerated Type

This field specifies the type of memory tiling (Linear, WMajor, XMajor, or YMajor) employed to tile this surface. See Memory Interface Functions for details on memory tiling and restrictions.

Value	Name	Description
0h	TILEMODE_LINEAR	Linear mode (no tiling)
1h	Reserved	Reserved
2h	TILEMODE_XMAJOR	X major tiling
3h	TILEMODE_YMAJOR	Y major tiling

Programming Notes

- Refer to *Memory Data Formats* for restrictions on TileMode direction for the various buffer types. (Of particular interest is the fact that YMAJOR tiling is not supported for display/overlay buffers).
- The corresponding cache(s) must be invalidated before a previously accessed surface is accessed again with an altered state of this field.
- Linear surfaces can be mapped to Main Memory (uncached) or System Memory (cacheable, snooped). Tiled (X/Y/W) surfaces can only be mapped to Main Memory.

MEDIA_SURFACE_STATE

3	31:30	**Reserved** Format: MBZ
	29:16	**X Offset for U(Cb)** Format: U14 Pixel Offset **Description** For non planar surfaces this field specifies the horizontal offset in pixels from the Surface Base Address to the start (origin) of the surface. For Planar surfaces this field specifies the horizontal offset in pixels from the Y-plane origin to the start (origin) of the U(Cb) plane or the interleaved UV plane if Interleave Chroma is enabled. Resultant X-offset = 'X-offset of the surface (Y-plane)' + 'X offset for U(Cb)' For TileYS and TileYF this offset should be integral multiple of Tile width of Luma 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
	13:0	**Y Offset for U(Cb)** Format: U14 Row Offset **Description** For non planar surfaces this field specifies the vertical offset in pixels from the Surface Base Address to the start (origin) of the surface. For Planar surfaces this field specifies the vertical offset in rows from the Y-plane origin to the start (origin) of the U(Cb) plane or the interleaved UV plane if Interleave Chroma is enabled. Resultant X-offset = 'Y-offset of the surface (Y-plane)' + 'Y offset for U(Cb)' For TileYS and TileYF this offset should be integral multiple of Tile width of Luma plane. **Programming Notes** This field must be aligned by 4 bit[1:0] = 00 This field must be aligned by 4 bit[1:0] = 00 for all format besides PLANAR_420_*
4	31:30	**Reserved** Format: MBZ
	29:16	**X Offset for V(Cr)** Exists If: //([Surface Format] is one of planar) AND ([Interleave Chroma] == '0') Format: U14 Pixel Offset

MEDIA_SURFACE_STATE

DWord	Bit	Description
		Description For Planar surfaces this field specifies the horizontal offset in pixels from the Y-plane origin to the start (origin) of the V(Cb) plane. Resultant X-offset = 'X-offset of the surface (Y-plane)' + 'X offset for V(Cb)' For TileYS and TileYF this offset should be integral multiple of Tile width of Luma plane. **Programming Notes** For PLANAR_420 and PLANAR_422 surface formats, this field must indicate an even number of pixels.
	15	**Reserved** Format: MBZ
	14:0	**Y Offset for V(Cr)** Exists If: //([Surface Format] is one of planar) AND ([Interleave Chroma] == '0') Format: U15 Row Offset **Description** For Planar surfaces this field specifies the vertical offset in rows from the Y-plane origin to the start (origin) of the V(Cb) plane. Resultant Y-offset = 'Y-offset of the surface (Y-plane)' + 'Y offset for V(Cb)' For TileYS and TileYF this offset should be integral multiple of Tile width of Luma plane. **Programming Notes** This field must indicate a multiple of 4 (bit 0 & 1 = 00).
5	31	**Vertical Line Stride** Format: U1 in lines to skip between logically adjacent lines For Surfaces accessed via the sample_8x8 message:Specifies number of lines (0 or 1) to skip between logically adjacent lines - provides support of interleaved (field) surfaces as textures.For Other Surfaces:Vertical Line Stride must be zero.
	30	**Vertical Line Stride Offset** Format: U1 in lines of initial offset (when Vertical Line Stride == 1) For Surfaces accessed via the sample_8x8 message: Specifies the offset of the initial line from the beginning of the buffer. For Other Surfaces: Vertical Line Stride Offset must be zero. **Programming Notes** This field must be set to 0 if Vertical Line Stride is 0.
	29:24	**Reserved** Format: MBZ
	23:20	**Reserved**

MEDIA_SURFACE_STATE

Format:	MBZ

19:18 Tiled Resource Mode

Format:	U2

For Sampling Engine, Render Target, and Typed/Untyped Surfaces:

This field specifies the tiled resource mode.

For other surfaces:

This field is ignored.

Value	Name	Description
0h	TRMODE_NONE	No tiled resource
1h	TRMODE_TILEYF	4KB tiled resources
2h	TRMODE_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.
If this field is not set to TRMODE_NONE, the **Surface Format** must be one with 8, 16, 32, 64, or 128 bits per element, or one of the compressed texture modes (BC*, ETC*, EAC*, ASTC*). Additionally, YCRCB* formats are supported and treated as 16 bits per element, and the PLANAR_420_8 and PLANAR_422_8 formats are supported and treated as 8 bits per element on the Y plane and 16 bits per element on the UV plane (if **Interleave Chroma** is enabled) or 8 bits per element on the U and V planes (if **Interleave Chroma** is disabled.

17:7 Reserved

Format:	MBZ

6:0 Surface Memory Object Control State

Default Value:	0h DefaultVaueDesc
Format:	MEMORY_OBJECT_CONTROL_STATE

This 7-bit field is used in various state commands and indirect state objects to define cacheability and other attributes related to memory objects.

6 — 31:0 Surface Base Address

Format:	GraphicsAddress[31:0]

Specifies the low 32 bits of the byte-aligned base address of the surface.

Programming Notes
For SURFTYPE_BUFFER render targets, this field specifies the base address of first element of the surface. The surface is interpreted as a simple array of that single element type. The address must be naturally-aligned to the element size (e.g., a buffer containing R32G32B32A32_FLOAT elements must be 16-byte aligned).For SURFTYPE_BUFFER non-rendertarget surfaces, this field specifies the base address of the first element of the surface, computed in software by adding the surface base address to the byte offset of the element in the buffer.Mipmapped, cube and 3D sampling engine surfaces are stored in a 'monolithic' (fixed) format, and only require a

MEDIA_SURFACE_STATE

		single address for the base texture.Linear render target surface base addresses must be element-size aligned, for non-YUV surface formats, or a multiple of 2 element-sizes for YUV surface formats. Other linear surfaces have no alignment requirements (byte alignment is sufficient.)Linear depth buffer surface base addresses must be 64-byte aligned. Note that while render targets (color) can be SURFTYPE_BUFFER, depth buffers cannot.Tiled surface base addresses must be 4KB-aligned. Note that only the offsets from Surface Base Address are tiled, Surface Base Address itself is not transformed using the tiling algorithm.For tiled surfaces, the actual start of the surface can be offset from the Surface Base Address by the X Offset and Y Offset fields.Certain message types used to access surfaces have more stringent alignment requirements. Please refer to the specific message documentation for additional restrictions.
7	31:16	**Reserved** Format: MBZ
	15:0	**Surface Base Address High** Format: GraphicsAddress[47:32] Specifies the high 16 bits of the byte-aligned base address of the surface. Refer to Surface Base Address [31:0] for programming notes applying to this field.

Power Clock State Format

<table>
<tr><th colspan="3">Power Clock State Format</th></tr>
<tr><td colspan="3">Source: RenderCS
Size (in bits): 31
Default Value: 0x00000266</td></tr>
<tr><td colspan="3">Known Uses
• R_PWR_CLK_STATE - Render Power Clock State Register
• PM_PWR_CLK_STATE - PM Power Clock State Request (Intended, in GT/GTI space, not yet in use)
• PM_PWR_CLK_STATE (Intended, in GT/GTI space, not yet in use)</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>30:19</td><td><b>RSVD</b>
<table><tr><td>Access:</td><td>RO</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table>
Reserved (CSunit implements full 32b storage)</td></tr>
<tr><td></td><td>18:15</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>14:13</td><td><b>RSVD</b>
<table><tr><td>Access:</td><td>RO</td></tr></table>
Reserved (CSunit implements full 32b storage)</td></tr>
<tr><td></td><td>12</td><td><b>Spare</b>
<table><tr><td>Access:</td><td>R/W</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table>
Spare bit</td></tr>
<tr><td></td><td>11</td><td><b>SSCountEn</b>
<table><tr><td>Access:</td><td>R/W</td></tr></table>
Enable Subslice Count Request.
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Disable</td><td>Use async PMunit subslice count request.</td></tr>
<tr><td>1h</td><td>Enable</td><td>Use SliceCount from this register.</td></tr></table></td></tr>
<tr><td></td><td>10:8</td><td><b>SScount</b>
<table><tr><td>Access:</td><td>R/W</td></tr></table>
Number of subslices to power.
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>001b</td><td></td><td>1 subslice.</td></tr>
<tr><td>010b</td><td></td><td>2 subslices.</td></tr>
<tr><td>100b</td><td><b>[Default]</b></td><td>3 subslices.</td></tr></table></td></tr>
</table>

Power Clock State Format

7:4 **EUmax**

Access:	R/W

Maximum number of EUs to power (per subslice if multiple subslices enabled).
To specify an exact number of subslices, set EUmax equal to EUmin.

Value	Name	Description
0010b		2 EUs
0100b		4 EUs
0110b	**[Default]**	6 EUs

Programming Notes
EUmin and EUmax need to be even and odd numbers are illegal; hardware will clip odd EU counts to an even value.

3:0 **EUmin**

Access:	R/W

Minimum number of EUs to power (per subslice if multiple subslices enabled).

Value	Name	Description
0010b		2 EUs
0100b		4 EUs
0110b	**[Default]**	6 EUs

Programming Notes
EUmin and EUmax need to be even and odd numbers are illegal; hardware will clip odd EU counts to an even value.

RENDER_SURFACE_STATE

RENDER_SURFACE_STATE

Source:	BSpec
Exists If:	//[MessageType] != 'Sample_8x8'
Size (in bits):	512
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

This is the normal surface state used by all messages that use SURFACE_STATE except those that use MEDIA_SURFACE_STATE.

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

This field defines the type of the surface.

Value	Name	Description
0h	SURFTYPE_1D	Defines a 1-dimensional map or array of maps
1h	SURFTYPE_2D	Defines a 2-dimensional map or array of maps
2h	SURFTYPE_3D	Defines a 3-dimensional (volumetric) map
3h	SURFTYPE_CUBE	Defines a cube map or array of cube maps
4h	SURFTYPE_BUFFER	Defines an element in a buffer
5h	SURFTYPE_STRBUF	Defines a structured buffer surface
6h	Reserved	
7h	SURFTYPE_NULL	Defines a null surface

Programming Notes

A null surface is used in instances where an actual surface is not bound. When a write message is generated to a null surface, no actual surface is written to. When a read message (including any sampling engine message) is generated to a null surface, the result is all zeros. Note that a null surface type is allowed to be used with all messages, even if it is not specificially indicated as supported. All of the remaining fields in surface state are ignored for null surfaces, with the following exceptions:

- **Width, Height, Depth, LOD,** and **Render Target View Extent** fields must match the depth buffer's corresponding state for all render target surfaces, including null.

All sampling engine and data port messages support null surfaces with the above behavior, even if not mentioned as specifically supported, except for the following:

- Data Port Media Block Read/Write messages
- Data Port Transpose Read message
- The **Surface Type** of a surface used as a render target (accessed via the Data Port's Render Target Write message) must be the same as the **Surface Type** of all other render targets and of the depth buffer (defined in 3DSTATE_DEPTH_BUFFER), unless either the depth buffer

RENDER_SURFACE_STATE

		or render targets are SURFTYPE_NULL.
	28	**Surface Array** Format: | Enable This field, if enabled, indicates that the surface is an array. **Programming Notes** If this field is *enabled*, the **Surface Type** must be SURFTYPE_1D, SURFTYPE_2D, or SURFTYPE_CUBE. If this field is *disabled* and **Surface Type** is SURFTYPE_1D, SURFTYPE_2D, or SURFTYPE_CUBE, the **Depth** field must be set to zero.
	27	**ASTC_Enable** Format: | Enable This field, if enabled, indicates that the surface is one of ASTC compression formats. **Programming Notes** If this field is *enabled*, the definition of **Surface Format** encoding will follow a new convention defined by ASTC. If this field is *disabled*, the definition of **Surface Format** will follow the legacy convention defined in non-ASTC style.
	26:18	**Surface Format** Format: | SURFACE_FORMAT **Description** This field specifies the format of the surface or element within this surface. This field is ignored for all data port messages other than the render target message and streamed vertex buffer write message. Some forms of the media block messages use the surface format. If **ASTC_Enable** is set to 0, the supported formats and their encoding is listed in the table (x) in Section (y); Otherwise the supported formats and their encoding is listed in the table (x+1) in Section (y). **Programming Notes** If **ASTC_Enable** is set to 0: YUV (YCRCB) surfaces used as render targets can only be rendered to using 3DPRIM_RECTLIST with even X coordinates on all of its vertices, and the pixel shader cannot kill pixels. If **Number of Multisamples** is set to a value other than MULTISAMPLECOUNT_1, this field cannot be set to the following formats: • Any compressed texture format (BC*, DXT*, FXT*, ETC*, EAC*) • Any YCRCB* format This field cannot ASTC format if the **Surface Type** is SURFTYPE_BUFFER or SURFTYPE_STRBUF This field cannot be ASTC format if the **Surface Type** is SURFTYPE_1D.

RENDER_SURFACE_STATE

This field cannot be a YUV (YCRCB*) or compressed (BC*, DXT*, FXT*, ETC*, EAC*) format if the **Surface Type** is SURFTYPE_BUFFER or SURFTYPE_STRBUF This field cannot be a planar YUV (PLANAR_*) or compressed (BC*, DXT*, FXT*, ETC*, EAC*) format if the **Surface Type** is SURFTYPE_1D.

17:16 **Surface Vertical Alignment**

Description
For Sampling Engine and Render Target Surfaces: This field specifies the vertical alignment requirement in elements for the surface. Refer to the "Memory Data Formats" chapter for details on how this field changes the layout of the surface in memory. An *element* is defined as a pixel in uncompresed surface formats, and as a compression block in compressed surface formats. For MSFMT_DEPTH_STENCIL type multisampled surfaces, an element is a sample.
This field is used for 2D, CUBE, and 3D surface alignment when Tiled Resource Mode is TRMODE_NONE (Tiled Resource Mode is disabled). This field is ignored for 1D surfaces and also when Tiled Resource Mode is not TRMODE_NONE (e.g. Tiled Resource Mode is enabled). See the appropriate Alignment table in the "Surface Layout and Tiling" section under Common Surface Formats for the table of alignment values for Tiled Resrouces.
For other surfaces: This field is ignored.

Value	Name	Description
0h	Reserved	Reserved
1h	VALIGN 4	Vertical alignment factor j = 4
2h	VALIGN 8	Vertical alignment factor j = 8
3h	VALIGN 16	Vertical alignment factor j = 16

Programming Notes
This field is intended to be set to VALIGN_4 if the surface was rendered as a depth buffer, for a multisampled (4x) render target, or for a multisampled (8x) render target, since these surfaces support only alignment of 4. Use of VALIGN_4 for other surfaces is supported, but increases memory usage.
This field is intended to be set to VALIGN_8 only if the surface was rendered as a stencil buffer, since stencil buffer surfaces support only alignment of 8. If set to VALIGN_8, Surface Format must be R8_UINT.
For uncompressed surfaces, the units of "j" are rows of pixels on the physical surface. For compressed texture formats, the units of "j" are in compression blocks, thus each increment in "j" is equal to h pixels, where h is the height of the compression block in pixels.

15:14 **Surface Horizontal Alignment**

Description
For Sampling Engine and Render Target Surfaces: This field specifies the horizontal alignment requirement for the surface.

RENDER_SURFACE_STATE

This field is used for alignment when LOD >= Mip Tail Start LOD
This field is ignored when Tiled Resource Mode is not TRMODE_NONE (i.e. Tiled Resources are enabled). See the "Surface Layout and Tiling" section under Common Surface Formats for the table of alignment values for Tile Resrouces.

For other surfaces:
This field is ignored.

Value	Name	Description
0h	Reserved	Reserved
1h	HALIGN 4	Horizontal alignment factor j = 4
2h	HALIGN 8	Horizontal alignment factor j = 8
3h	HALIGN 16	Horizontal alignment factor j = 16

Programming Notes
This field is intended to be set to HALIGN_8 only if the surface was rendered as a depth buffer with Z16 format or a stencil buffer. In this case it must be set to HALIGN_8 since these surfaces support only alignment of 8. For Z32 formats it must be set ot HALIGN_4. Use of HALIGN_8 for other surfaces is supported, but increases memory usage. For uncompressed surfaces, the units of "i" are pixels on the physical surface. For compressed texture formats, the units of "i" are in compression blocks, thus each increment in "i" is equal to w pixels, where w is the width of the compression block in pixels.
When Auxiliary Surface Mode is set to AUX_CCS_D or AUX_CCS_E, HALIGN 16 must be used.

13:12 **Tile Mode**

This field specifies the type of memory tiling (Linear, WMajor, XMajor, or YMajor) employed to tile this surface. See *Memory Interface Functions* for details on memory tiling and restrictions.

Value	Name	Description
0h	LINEAR	Linear mode (no tiling)
1h	WMAJOR	W major tiling
2h	XMAJOR	X major tiling
3h	YMAJOR	Y major tiling

Programming Notes

- Refer to *Memory Data Formats* for restrictions on *TileMode* direction for the various buffer types. (Of particular interest is the fact that YMAJOR tiling is not supported for display/overlay buffers).
- The corresponding cache(s) must be invalidated before a previously accessed surface is accessed again with an altered state of this field.
- Use of WMAJOR is valid only for sampling engine, Data Cache Data Port and render target surfaces and **Surface Format** must be R8_UINT. Vertical Line Stride must be zero. In addition to W tiling, this mode implies that the surface is stored as a stencil buffer. Refer to

<table>
<tr><th colspan="3">RENDER_SURFACE_STATE</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td><i>Memory Data Formats</i> section for details on stencil buffer surface layout.<ul><li>Linear surfaces can be mapped to Main Memory (uncached) or System Memory (cacheable, snooped). Tiled (X/Y/W) surfaces can only be mapped to Main Memory.</li><li>If <b>Surface Type</b> is SURFTYPE_BUFFER, this field must be TILEMODE_LINEAR</li><li>If <b>Number of Multisamples</b> is not MULTISAMPLECOUNT_1, this field must be YMAJOR.</li></ul></td></tr>
<tr><td>If <b>Surface Type</b> is SURFTYPE_STRBUF, this field must be TILEMODE_LINEAR.</td></tr>
<tr><td>If <b>Surface Type</b> is SURFTYPE_1D this field must be TILEMODE_LINEAR, unless <b>Sampler Legacy 1D Map Layout Disable</b> is set to 0, in which case TILEMODE_YMAJOR and TILEMODE_WMAJOR are also allowed. <b>Tiled Resource Mode</b> must be set to TRMODE_NONE for these cases.</td></tr>
<tr><td>TILEMODE_XMAJOR is only allowed if Surface Type is SURFTYPE_2D.</td></tr>
<tr><td>If <b>Surface Format</b> is ASTC*, this field must be TILEMODE_YMAJOR.</td></tr>
</table>
</td></tr>
<tr><td></td><td>11</td><td>
<b>Vertical Line Stride</b>
<table><tr><td>Format:</td><td>U1 In lines to skip between logically adjacent lines</td></tr></table>
<b>For 2D Non-Array Surfaces accessed via the Sampling Engine or Data Cache Data Port:</b> Specifies number of lines (0 or 1) to skip between logically adjacent lines - provides support of interleaved (field) surfaces as textures.
<b>For Other Surfaces:</b> Vertical Line Stride must be zero.
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>This bit must not be set if the surface format is a compressed type (BCn*, FXT1, ETC*, EAC*).</td></tr>
<tr><td>This bit must not be set if the surface format is compressed type ASTC*.</td></tr>
<tr><td>This bit must not be set if the <b>Auxiliary Surface Mode</b> is not AUX_NONE.</td></tr>
<tr><td>If this bit is set on a sampling engine surface, the mip mode filter must be set to MIPFILTER_NONE and the min and mag mode filter cannot be set to MAPFILTER_FLEXIBLE.</td></tr>
</table>
</td></tr>
<tr><td></td><td>10</td><td>
<b>Vertical Line Stride Offset</b>
<table><tr><td>Format:</td><td>U1 In lines of initial offset (when Vertical Line Stride == 1)</td></tr></table>
<b>For 2D Non-Array Surfaces accessed via the Sampling Engine or Data Cache Data Port:</b>
Specifies the offset of the initial line from the beginning of the buffer. Ignored when Vertical <b>Line Stride</b> is 0.
<b>For Other Surfaces:</b>
Vertical Line Stride Offset must be zero.
</td></tr>
<tr><td></td><td>9</td><td>
<b>Sampler L2 Out of Order Mode Disable</b>
<table><tr><td>Format:</td><td>Disable</td></tr></table>
If disabled this will forced formats which would have bypassed the L2 and been filled into the L1 out of order to be cached in the L2 and send in order to the L1. In general that is any format which is expaned 1:2 in L1 or not expanded at all. This would include all lossless compressed cases
For all other formats this will have no affect.
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>This bit must be set for the following surface types: BC2_UNORM BC3_UNORM BC5_UNORM BC5_SNORM BC7_UNORM</td></tr>
</table>
</td></tr>
</table>

RENDER_SURFACE_STATE

8

Render Cache Read Write Mode

For Surfaces accessed via the Data Port to Render Cache:

This field specifies the way Render Cache treats a write request. If unset, Render Cache allocates a write-only cache line for a write miss. If set, Render Cache allocates a read-write cache line for a write miss.

For Surfaces accessed via the Sampling Engine or Data Port to Texture Cache or Data Cache:

This field is reserved : MBZ

Value	Name	Description
0h	Write-Only Cache	Allocating write-only cache for a write miss
1h	Read-Write Cache	Allocating read-write cache for a write miss

Programming Notes
This field is provided for performance optimization for Render Cache read/write accesses (from Gen4 EU's point of view).

7:6

Media Boundary Pixel Mode

For 2D Non-Array Surfaces accessed via the Data Port Media Block Read Message or Data Port Transpose Read message:

This field enables control of which rows are returned on vertical out-of-bounds reads using the Data Port Media Block Read Message or Data Port Transpose Read message. In the description below, frame mode refers to **Vertical Line Stride** = 0, field mode is **Vertical Line Stride** = 1 in which only the even or odd rows are addressable. The frame refers to the entire surface, while the field refers only to the even or odd rows within the surface.

For Other Surfaces:

Reserved : MBZ

Value	Name	Description
0h	NORMAL_MODE	The row returned on an out-of-bound access is the closest row in the frame or field. Rows from the opposite field are never returned.
1h	Reserved	
2h	PROGRESSIVE_FRAME	The row returned on an out-of-bound access is the closest row in the frame, even if in field mode.
3h	INTERLACED_FRAME	In field mode, the row returned on an out-of-bound access is the closest row in the field. In frame mode, even out-of-bound rows return the nearest even row while odd out-of-bound rows return the nearest odd row.

5

Cube Face Enable - Negative X

Exists If:	[Surface Type] == 'SURFTYPE_CUBE'
Format:	Enable

For SURFTYPE_CUBE Surfaces accessed via the Sampling Engine:

This field enable the individual face of a cube map. Enabling a face indicates that the face is present in the cube map, while disabling it indicates that that face is represented by the texture map's border color. Refer to Memory Data Formats for the correlation between faces and the cube map

RENDER_SURFACE_STATE

		memory layout. Note that storage for disabled faces must be provided. **Programming Notes** When TEXCOORDMODE_CLAMP is used when accessing a cube map, this field must be programmed to 1b (face enabled).
	4	**Cube Face Enable - Positive X** Exists If: | [Surface Type] == 'SURFTYPE_CUBE' Format: | Enable **For SURFTYPE_CUBE Surfaces accessed via the Sampling Engine:** This field enable the individual face of a cube map. Enabling a face indicates that the face is present in the cube map, while disabling it indicates that that face is represented by the texture map's border color. Refer to Memory Data Formats for the correlation between faces and the cube map memory layout. Note that storage for disabled faces must be provided. **Programming Notes** When TEXCOORDMODE_CLAMP is used when accessing a cube map, this field must be programmed to 1b (face enabled).
	3	**Cube Face Enable - Negative Y** Exists If: | [Surface Type] == 'SURFTYPE_CUBE' Format: | Enable **For SURFTYPE_CUBE Surfaces accessed via the Sampling Engine:** This field enable the individual face of a cube map. Enabling a face indicates that the face is present in the cube map, while disabling it indicates that that face is represented by the texture map's border color. Refer to Memory Data Formats for the correlation between faces and the cube map memory layout. Note that storage for disabled faces must be provided. **Programming Notes** When TEXCOORDMODE_CLAMP is used when accessing a cube map, this field must be programmed to 1b (face enabled).
	2	**Cube Face Enable - Positive Y** Exists If: | [Surface Type] == 'SURFTYPE_CUBE' Format: | Enable **For SURFTYPE_CUBE Surfaces accessed via the Sampling Engine:** This field enable the individual face of a cube map. Enabling a face indicates that the face is present in the cube map, while disabling it indicates that that face is represented by the texture map's border color. Refer to Memory Data Formats for the correlation between faces and the cube map memory layout. Note that storage for disabled faces must be provided. **Programming Notes** When TEXCOORDMODE_CLAMP is used when accessing a cube map, this field must be programmed to 1b (face enabled).

<table>
<tr><th colspan="3">RENDER_SURFACE_STATE</th></tr>
<tr><td></td><td>1</td><td><b>Cube Face Enable - Negative Z</b>
<table>
<tr><td>Exists If:</td><td>[Surface Type] == 'SURFTYPE_CUBE'</td></tr>
<tr><td>Format:</td><td>Enable</td></tr>
</table>
<b>For SURFTYPE_CUBE Surfaces accessed via the Sampling Engine:</b>
This field enable the individual face of a cube map. Enabling a face indicates that the face is present in the cube map, while disabling it indicates that that face is represented by the texture map's border color. Refer to Memory Data Formats for the correlation between faces and the cube map memory layout. Note that storage for disabled faces must be provided.
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>When TEXCOORDMODE_CLAMP is used when accessing a cube map, this field must be programmed to 1b (face enabled).</td></tr>
</table>
</td></tr>
<tr><td></td><td>5:0</td><td><b>Reserved</b>
<table>
<tr><td>Exists If:</td><td>[Surface Type] != 'SURFTYPE_CUBE'</td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
</table>
</td></tr>
<tr><td></td><td>0</td><td><b>Cube Face Enable - Positive Z</b>
<table>
<tr><td>Exists If:</td><td>[Surface Type] == 'SURFTYPE_CUBE'</td></tr>
<tr><td>Format:</td><td>Enable</td></tr>
</table>
<b>For SURFTYPE_CUBE Surfaces accessed via the Sampling Engine:</b>
This field enable the individual face of a cube map. Enabling a face indicates that the face is present in the cube map, while disabling it indicates that that face is represented by the texture map's border color. Refer to Memory Data Formats for the correlation between faces and the cube map memory layout. Note that storage for disabled faces must be provided.
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>When TEXCOORDMODE_CLAMP is used when accessing a cube map, this field must be programmed to 1b (face enabled).</td></tr>
</table>
</td></tr>
<tr><td>1</td><td>31</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table>
</td></tr>
<tr><td></td><td>30:24</td><td><b>Memory Object Control State</b>
<table>
<tr><td>Format:</td><td>MEMORY_OBJECT_CONTROL_STATE</td></tr>
</table>
Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).
</td></tr>
</table>

RENDER_SURFACE_STATE

23:19 **Base Mip Level**

Format:	U4.1

Range: [0.0, 14.0]
Specifies which mip level is considered the "base" level when determining mag-vs-min filter and selecting the "base" mip level.

Programming Notes
This field also exists in SAMPLER_STATE. If both fields are zero, the Base Mip Level is zero. If one is nonzero, Base Mip Level is the nonzero field. It is illegal to have both Base Mip Level fields nonzero.

18 **Reserved**

Format:	MBZ

17 **Reserved**

Format:	MBZ

16 **Reserved**

Format:	MBZ

15 **Reserved**

Format:	MBZ

14:0 **Surface QPitch**

Format:	QPitch[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. For Quilted Textures this field specifies the distance in rows between *quilt* slices. For compressed texture formats, one row contains a complete compression block vertically.
- SURFTYPE_3D: distance in *rows* between R-slices [**Note:** these *rows* are only in the vertical dimension without considering the depth dimension]. For compressed texture formats, one row contains a complete compression block vertically.
- Other surface types: field is ignored

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

Programming Notes
For Surface Type 1D: This field must be set to an integer multiple of the **Surface Horizontal**

<table>
<tr><th colspan="3">RENDER_SURFACE_STATE</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td><b>Alignment</b>
<b>For Surface Type 2D, CUBE:</b> This field must be set to an integer multiple of the <b>Surface Vertical Alignment</b>
<b>For Surface Type 3D:</b> <i>Tile Mode != Linear:</i> This field must be set to an integer multiple of the tile height (2^Cv) <i>Tile Mode == Linear:</i> This field must be set to an integer multiple of the Surface Vertical Alignment
<b>Note:</b> for compressed textures (BC*, FXT1, ETC*, EAC*), this field is in units of rows of compression blocks.</td></tr>
<tr><td><b>Note:</b> for the compressed texture ASTC Surface Format, this field is in units of rows of compression blocks.</td></tr>
<tr><td>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.</td></tr>
</table>
</td></tr>
<tr><td>2</td><td>31:30</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table>
</td></tr>
<tr><td></td><td>29:16</td><td><b>Height</b>
<table>
<tr><td>Format:</td><td>U14-1</td></tr>
</table>
This field specifies the height of the surface, minus 1. If the surface is MIP-mapped, this field contains the height of the base MIP level. For buffers, this field specifies a portion of the buffer size.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th><th>Exists If</th></tr>
<tr><td>[0,0]</td><td></td><td>must be zero</td><td>[Surface Type] == 'SURFTYPE_1D'</td></tr>
<tr><td>[0,16383]</td><td></td><td>height of surface - 1 (y/v dimension)</td><td>[SurfaceType] == 'SURFTYPE_2D'</td></tr>
<tr><td>[0,16383]</td><td></td><td>height of surface - 1 (y/v dimension)</td><td>[SurfaceType] == 'SURFTYPE_3D'</td></tr>
<tr><td>[0,16383]</td><td></td><td>height of surface - 1 (y/v dimension)</td><td>[SurfaceType] == 'SURFTYPE_CUBE'</td></tr>
<tr><td>[0,16383]</td><td></td><td>contains bits [20:7] of the number of entries in the buffer - 1</td><td>([SurfaceType] == 'SURFTYPE_BUFFER') ||
([SurfaceType] == 'SURFTYPE_STRBUF')</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>For typed buffer and structured buffer surfaces, the number of entries in the buffer ranges from 1 to 2^{27}. For raw buffer surfaces, the number of entries in the buffer is the number of bytes which can range from 1 to 2^{30}. After subtracting one from the number of entries, software must place the fields of the resulting 27-bit value into the <b>Height, Width</b>, and <b>Depth</b> fields as indicated, right-justified in each field. Unused upper bits must be set to zero.</td></tr>
<tr><td>If <b>Vertical Line Stride</b> is 1, this field indicates the height of the field, not the height of the frame</td></tr>
<tr><td>The <b>Height</b> of a render target must be the same as the <b>Height</b> of the other render targets and the depth buffer (defined in 3DSTATE_DEPTH_BUFFER), unless <b>Surface Type</b> is SURFTYPE_1D or SURFTYPE_2D with <b>Depth</b> = 0 (non-array) and <b>LOD</b> = 0 (non-mip mapped).</td></tr>
</table>
</td></tr>
</table>

RENDER_SURFACE_STATE

If this surface in memory is accessed with Vertical Line Stride set to both 0 and 1, this field must be an even value when Vertical Line Stride is 0.
If Media Pixel Boundary Mode is not set to NORMAL_MODE, this field must be an even value.
If Surface Format is PLANAR*, see Planar Memory Organization section for restrictions on the value of this field.

15:14 **Reserved**

Format:	MBZ

13:0 **Width**

Format:	U14-1

Description
This field specifies the width of the surface, minus 1. If the surface is MIP-mapped, this field specifies the width of the base MIP level. The width is specified in units of pixels or texels. For buffers, this field specifies a portion of the buffer size.
For surfaces accessed with the Media Block Read/Write message, this field is in units of DWords.
For surfaces accessed with the Transpose Read Message, this field is in units of DWords.

Value	Name	Description	Exists If
[0,16383]		width of surface - 1 (x/u dimension)	[SurfaceType] == 'SURFTYPE_1D'
[0,16383]		width of surface - 1 (x/u dimension)	[SurfaceType] == 'SURFTYPE_2D'
[0,16383]		width of surface - 1 (x/u dimension)	[SurfaceType] == 'SURFTYPE_3D'
[0,16383]		width of surface - 1 (x/u dimension)	[SurfaceType] == 'SURFTYPE_CUBE'
[0,127]		contains bits [6:0] of the number of entries in the buffer - 1	([SurfaceType] == 'SURFTYPE_BUFFER') || ([SurfaceType] == 'SURFTYPE_STRBUF')

Programming Notes

- For surface types other than SURFTYPE_BUFFER or STRBUF 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 the Height.
- For MONO8 textures, Width must be a multiple of 32 texels.
- The **Width** of a render target must be the same as the **Width** of the other render target(s) and the depth buffer (defined in 3DSTATE_DEPTH_BUFFER), unless **Surface Type** is SURFTYPE_1D or SURFTYPE_2D with **Depth** = 0 (non-array) and **LOD** = 0 (non-mip mapped).

RENDER_SURFACE_STATE

- The **Width** of a render target with YUV surface format must be a multiple of 2.
- For SURFTYPE_BUFFER: The low two bits of this field must be 11 if the Surface Format is RAW (the size of the buffer must be a multiple of 4 bytes).

If **Surface Format** is PLANAR*, this field must be a multiple of 2

If **Number of Multisamples** is MULTISAMPLECOUNT_16, then Width must be 8K texels or less, or the surface must not use the a multisample control surface (MCS).

3 | 31:21 | **Depth**

Format:	U11-1

This field specifies the total number of levels, minus 1, for a volume texture or the number of array elements, minus 1, 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. For buffers, this field specifies a portion of the buffer size.

Value	Name	Description	Exists If
[0,2047]		number of array elements - 1	[SurfaceType] == 'SURFTYPE_1D'
[0,2047]		number of array elements - 1	[SurfaceType] == 'SURFTYPE_2D'
[0,2047]		depth of surface - 1 (z/r dimension)	[SurfaceType] == 'SURFTYPE_3D'
[0,340]		number of array elements - 1 [see programming notes for range]	[SurfaceType] == 'SURFTYPE_CUBE'
[0,511]		contains bits [29:21] of the number of entries in the buffer - 1	([SurfaceType] == SURFTYPE_BUFFER) OR ([SurfaceType] == 'SURFTYPE_STRBUF')

Programming Notes
The **Depth** of a render target must be the same as the **Depth** of the other render target(s) and of the depth buffer (defined in 3DSTATE_DEPTH_BUFFER).
For SURFTYPE_CUBE: For Sampling Engine Surfaces and Typed Data Port Surfaces, the range of this field is [0,340], indicating the number of cube array elements (equal to the number of underlying 2D array elements divided by 6). For other surfaces, this field must be zero.
For SURFTYPE_1D, 2D, and CUBE: The range of this field is reduced by one for each increase from zero of **Minimum Array Element**. For example, if **Minimum Array Element** is set to 1024 on a 2D surface, the range of this field is reduced to [0,1023].

20 | **Reserved**

Format:	MBZ

19 | **Reserved**

Format:	MBZ

RENDER_SURFACE_STATE

18 **Reserved**

Format:	MBZ

17:0 **Surface Pitch**

Format:	U18-1 Pitch in #Bytes

Surface Pitch Range:

- For surfaces of type SURFTYPE_BUFFER: [0,2047] -> [1B, 2048B]
- For surfaces of type SURFTYPE_STRBUF: [0,2047] -> [1B, 2048B]
- For other linear surfaces: [0, 262143] -> [1B, 256KB]
- For X-tiled surface: [511, 262143] -> [512B, 256KB] = [1 tile, 512 tiles]
- For Y-tiled surfaces: [127, 262143]->[128B, 256KB] = [1 tile, 2048 tiles]
- For W-tiled surfaces: [127, 262143]->[128B, 256KB] = [1 tile, 2048 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}-1,262143] -> [(2^{Cu})B,256KB] = [1 tile, 256KB/(2^{Cu}) tiles]

This field specifies the surface pitch in (#Bytes - 1).
For surfaces of type SURFTYPE_BUFFER and SURFTYPE_STRBUF, this field indicates the size of the structure.

Programming Notes
• For linear *render target* surfaces and surfaces accessed with the typed data port messages, the pitch must be a multiple of the element size for non-YUV surface formats. Pitch must be a multiple of 2 * element size for YUV surface formats. • For untyped data port messages, which are only supported with **Surface Type** SURFTYPE_BUFFER, the pitch is ignored and assumed to be 1 byte. • For linear surfaces with **Surface Type** of SURFTYPE_STRBUF, the pitch must be a multiple of 4 bytes. • For linear surfaces with **Surface Type** of SURFTYPE_BUFFER and **Surface Format** RAW, the pitch must be 1 byte. • For other linear surfaces, the pitch can be any multiple of bytes. • For tiled surfaces, the pitch must be a multiple of the tile width.
If the surface is a stencil buffer (and thus has **Tile Mode** set to TILEMODE_WMAJOR), 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).
• The width of a tile depends on the surface format if Tiled Resource Enable is enabled. Refer to the Tiled Resource Enable field to determine which sub-mode applies to the surface format in use, and determine the Cu parameter from the Surface Layout section. The tile width is equal to 2^Cu bytes. • For surfaces of type SURFTYPE_1D, this field is ignored.

RENDER_SURFACE_STATE

The following table indicates the maximum byte width, frame width, and pitch size allowed when memory compression is on.

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

4

31

Reserved

Exists If:	[Surface Type] != 'SURFTYPE_STRBUF'
Format:	MBZ

30:29

Render Target And Sample Unorm Rotation

Exists If:	[Surface Type] != 'SURFTYPE_STRBUF'

Description
For Render Target Surfaces: This field specifies the rotation of this render target surface when being written to memory.
For sample_unorm Messages: This field specifies the rotation of the data returned by sampler for sample_unorm message.
For Other Surfaces: This field is ignored.

Value	Name	Description
0h	0DEG	No rotation (0 degrees)
1h	90DEG	Rotate by 90 degrees
2h	180DEG	Rotate by 180 degrees [for sample_unorm message]

RENDER_SURFACE_STATE

3h	270DEG	Rotate by 270 degrees

Programming Notes
Programming Notes for Render Target Surfaces only • Rotation is not supported for render targets of any type other than simple, non-mip-mapped, non-array 2D surfaces. The surface must be using tiled with X major. • **Width** and **Height** fields apply to the dimensions of the surface before rotation. • For 90 and 270 degree rotated surfaces, the **Height** (rather than the **Width**) must be less than or equal to the **Surface Pitch** (specified in bytes). • For 90 and 270 degree rotated surfaces, the actual **Height** and **Width** of the surface in pixels (not the field value which is decremented) must both be even.
Rotation is supported only for surfaces with the following surface formats: B5G6R5_UNORM, B5G6R5_UNORM_SRGB, R8G8B8A8_UNORM, R8G8B8A8_UNORM_SRGB, B8G8R8[A|X]8_UNORM, B8G8R8[A|X]8_UNORM_SRGB, B10G10R10[A|X]2_UNORM, B10G10R10A2_UNORM_SRGB, R10G10B10A2_UNORM, R10G10B10A2_UNORM_SRGB, R16G16B16A16_FLOAT, R16G16B16X16_FLOAT

28:18 **Minimum Array Element**

Exists If:	[Surface Type] != 'SURFTYPE_STRBUF'
Format:	U11

17:7 **Render Target View Extent**

Exists If:	[Surface Type] != 'SURFTYPE_STRBUF'
Format:	U11-1

Range [0,2047] to indicate extent of [1,2048]

For Render Target and Typed Dataport 3D Surfaces:

This field indicates the extent of the accessible 'R' coordinates minus 1 on the LOD currently being rendered to.

For Render Target and Typed Dataport 1D and 2D Surfaces:

This field must be set to the same value as the Depth field.

For Other Surfaces:

This field is ignored.

6 **Multisampled Surface Storage Format**

Exists If:	[Surface Type] != 'SURFTYPE_STRBUF'

This field indicates the storage format of the multisampled surface.

Value	Name	Description
0h	MSS	Multisampled surface was/is rendered as a render target
1h	DEPTH_STENCIL	Multisampled surface was rendered as a depth or stencil buffer

RENDER_SURFACE_STATE

Programming Notes
• All multisampled render target surfaces must have this field set to MSFMT_MSS • IF this field is MSFMT_DEPTH_STENCIL, the only sampling engine messages allowed are "ld2dms", "resinfo", and "sampleinfo". • This field is ignored if **Number of Multisamples** is MULTISAMPLECOUNT_1

5:3 **Number of Multisamples**

Exists If:	[Surface Type] != 'SURFTYPE_STRBUF'

This field indicates the number of multisamples on the surface.

Value	Name
0h	MULTISAMPLECOUNT_1
1h	MULTISAMPLECOUNT_2
2h	MULTISAMPLECOUNT_4
3h	MULTISAMPLECOUNT_8
4h	MULTISAMPLECOUNT_16
5h-7h	Reserved

Programming Notes
If this field is any value other than MULTISAMPLECOUNT_1, the **Surface Type** must be SURFTYPE_2D This field must be set to MULTISAMPLECOUNT_1 unless the surface is a Sampling Engine surface or Render Target surface.

31:0 **Reserved**

Exists If:	[Surface Type] == 'SURFTYPE_STRBUF'
Format:	MBZ

2:0 **Multisample Position Palette Index**

Exists If:	[Surface Type] != 'SURFTYPE_STRBUF'

This field indicates the index into the sample position palette that the multisampled surface is using. This field is only used as a return value for the sampleinfo message, and is otherwise not used by hardware.

Value	Name
[0,7]	

5 31:25 **X Offset**

Format:	PixelOffset[8:2]

This field specifies the horizontal offset in pixels from the **Surface Base Address** to the start (origin) of the surface.

This field effectively loosens the alignment restrictions on the origin of tiled surfaces. Previously, tiled surface origin was (by definition) located at the base address, and thus needed to satisfy the 4KB base address alignment restriction. Now the origin can be specified at a finer (4-wide x 4-high pixel)

RENDER_SURFACE_STATE

resolution.

Value	Name	Description
[0,508]		In multiples of 4 (low 2 bits missing)

Programming Notes
• For linear surfaces, this field must be zero. • For surfaces accessed with the *Data Port Media Block Read/Write* message, the pixel size is assumed to be 32 bits in width. • For surfaces accessed with the **Data Port Transpose Read message**, the pixel size is assumed to be 32 bits in width. • For **Surface Format** with other than 8, 16, 32, 64, or 128 bits per pixel, this field must be zero. • If **Render Target Rotation** is set to other than RTROTATE_0DEG, this field must be zero. • If **Surface Type** not SURFTYPE_2D, this field must be zero. • If **MIP Count** is not zero, this field must be zero. • If **Number of Multisamples** is not MULTISAMPLECOUNT_1, this field must be zero. • If **Surface Array** is enabled, this field must be zero. • If **Auxiliary Surface Mode** is not AUX_NONE, this field must be zero. • If **Surface Vertical Alignment** is VALIGN_8, this field must be a multiple of 8. • For **Surface Format** with 8 bits per element, this field must be a multiple of 16. • For **Surface Format** with 16 bits per element, this field must be a multiple of 8.
• If **Tiled Resource Mode** is not TRMODE_NONE, this field must be zero.

24 **Reserved**

Format:	MBZ

23:21 **Y Offset**

Format:	RowOffset[4:2]

This field specifies the vertical offset in rows from the **Surface Base Address** to the start of the surface. (See additional description in the **X Offset** field.)

Value	Name	Description
[0,28]		In multiples of 4 (low two bits missing)

Programming Notes
• For linear surfaces, this field must be zero. • For render targets in which the **Render Target Array Index** is not zero, this field must be zero. • For **Surface Format** with other than 8, 16, 32, 64, or 128 bits per pixel, this field must be zero.

RENDER_SURFACE_STATE

- If **Render Target Rotation** is set to other than RTROTATE_0DEG, this field must be zero.
- If **Surface Type** not SURFTYPE_2D, this field must be zero.
- If **MIP Count** is not zero, this field must be zero.
- If **Number of Multisamples** is not MULTISAMPLECOUNT_1, this field must be zero.
- If **Surface Array** is enabled, this field must be zero.
- If **Auxiliary Surface Mode** is not AUX_NONE, this field must be zero.

- If **Tiled Resource Mode** is not TRMODE_NONE, this field must be zero.

This field must be zero if Surface Format is Planar and the U and V planes are half-pitch (e.g. YV12 format).

20

EWA Disable For Cube

Format:	Disable

Specifies if EWA mode for LOD quality improvement needs to be disabled for cube maps.

Value	Name	Description
0h	Enable **[Default]**	EWA is enabled for cube maps
1h	Disable	EWA is disabled for cube maps

Programming Notes
This field indicates if EWA mode for LOD quality improvement needs to be disabled for cube maps. By default EWA would be on for cube maps hence this field must be 0. If there is any spec violation seen with EWA on cube maps then this field must be set to 1 to disable EWA for cubes.

19:18

Tiled Resource Mode

For Sampling Engine, Render Target, and Typed/Untyped Surfaces:

This field specifies the tiled resource mode.

For other surfaces:

This field is ignored.

Value	Name	Description	Exists If
0h	NONE	No tiled resource	
1h	4KB	4KB tiled resources	[SurfaceType] == 'SURFTYPE_1D'
2h	64KB	64KB tiled resources	[SurfaceType] == 'SURFTYPE_1D'
1h	TILEYF	4KB tiled resources	[SurfaceType] != 'SURFTYPE_1D'
2h	TILEYS	64KB tiled resources	[SurfaceType] != 'SURFTYPE_1D'
3h	Reserved		

Programming Notes
If **Tile Mode** is not set to TILEMODE_YMAJOR, this field must be set to TRMODE_NONE, unless the Surface Type is SURFTYPE_1D.
If this field is not set to TRMODE_NONE, the **Surface Format** must be one with 8, 16, 32, 64, or 128

<table>
<tr><th colspan="3">RENDER_SURFACE_STATE</th></tr>
<tr><td></td><td></td><td><table>
<tr><td>bits per element, or one of the compressed texture modes (BC*, ETC*, EAC*, ASTC*). Additionally, YCRCB* formats are supported and treated as 16 bits per element, and the PLANAR_420_8 format is support and treated as 8 bits per element on the Y plane and 16 bits per element on the UV plane (if <b>Separate UV Plane Enable</b> is disabled) or 8 bits per element on the U and V planes (if <b>Separate UV Plane Enable</b> is enabled).</td></tr>
<tr><td>If this field is set to TRMODE_NONE, the surface cannot contain any null pages unless <b>Surface Type</b> is BUFFER or STRBUF. A BUFFER or STRBUF surface with null pages must have <b>Surface Base Address</b> and <b>Surface Pitch</b> set to an integer multiple of the element size, and <b>Surface Format</b> must be one with 8, 16, 32, 64, or 128 bits per element.</td></tr>
<tr><td>If <b>Surface Format</b> is PLANAR, the surface cannot contain any null pages.</td></tr>
</table></td></tr>
<tr><td></td><td>17:16</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>15</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>14</td><td><b>Coherency Type</b>
Specifies the type of coherency maintained for this surface.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>GPU coherent</td><td>Surface memory is kept coherent with GPU threads using GPU read/write ordering rules. Surface memory is backed by system memory but is not kept coherent with CPU (LLC).</td></tr>
<tr><td>1h</td><td>IA coherent</td><td>Surface memory is kept coherent with CPU (LLC).</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>This field may optionally be 1 (IA coherent) for messages sent to SFID_DP_DC0 or SFID_DP_DC1 or SFID_DP_DC2. This field is typically set to 0 (GPU coherent) if the context is operating in a non-SVM legacy mode (for example, Ring Buffer or a Execlist using 32-bit Virtual Address Legacy Context PPGTT32).</td></tr>
<tr><td>If the surface is not IA coherent, then it is possible that data written to Null Tile address will be cached, and then a later read to that same address might return a non-zero value. If a value is cached for a Null Tile address, the data will be eventually be discarded when the cache line is evicted. If the surface is IA coherent, then the cache line never contains a non-zero value, and Null Tile reads always return zero.</td></tr>
</table></td></tr>
<tr><td></td><td>13:12</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11:8</td><td><b>Mip Tail Start LOD</b><table><tr><td>Format:</td><td>U4 in LOD Units</td></tr></table>
<b>For Sampling Engine, Render Target, and Typed Surfaces:</b>
This field indicates which LOD is the first one in the MIP tail if <b>Tiled Resource Mode</b> is not TRMODE_NONE. The MIP tail has a different layout than the rest of the surface. Refer to the <i>Memory Data Formats</i> section for more details.</td></tr>
</table>

RENDER_SURFACE_STATE

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 Resource Mode** is TRMODE_NONE.
If **Tiled Resource 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.
If **Tiled Resource Mode** is not TRMODE_NONE, to disable the Mip Tail this field must be set to a mip that larger than those present in the surface (i.e. 15). This is recommended for non-mip-mapped surfaces.
The following table indicates the *maximum* size of the mip that is set to be the Mip Tail Start LOD for various cases:

Surface Type	Tiling Mode	#MS	Bits Per Element				
			8	16	32	64	128
1D	64KB	1	16384	8192	4096	2048	1024
	4KB	1	1024	512	256	128	64
2D/ CUBE	TIleYS	1	128x256	128x128	64x128	64x64	32x64
		2	128x128	128x64	64x64	64x32	32x32
		4	64x128	64x64	32x64	32x32	16x32
		8	64x64	64x32	32x32	32x16	16x16
		16	32x64	32x32	16x32	16x16	8x16
	TileYF	any	32x64	32x32	16x32	16x16	8x16
3D	TIleYS	1	32x32x32	16x32x32	16x32x16	16x16x16	8x16x16
	TIleYF	1	16x8x16	8x8x16	8x8x8	8x4x8	4x4x8

7:4

Surface Min LOD

Format:	U4 In LOD Units

For Sampling Engine and Typed Surfaces:
This field indicates the most detailed LOD that can be accessed as part of this surface. This field is added to the delivered LOD (*sample_l*, *ld*, or *resinfo* message types) before it is used to address the surface.

For Other Surfaces:
This field is ignored.

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

<table>
<tr><th colspan="3">RENDER_SURFACE_STATE</th></tr>
<tr><td></td><td>3:0</td><td><b>MIP Count / LOD</b>
<table>
<tr><td>Format:</td><td><b>Sampling Engine and Typed Surfaces:</b>
U4 in (LOD units - 1)
<b>Render Target Surfaces:</b>
U4 in LOD units</td></tr>
<tr><td>Range</td><td><b>Sampling Engine and Typed Surfaces:</b>
[0,14] representing [1,15] MIP levels
<b>Render Target Surfaces:</b> [0,14] representing LOD
<b>Other Surfaces:</b> [0]</td></tr>
</table>
<b>For Sampling Engine and Typed Surfaces:</b>
This field indicates the number of MIP levels allowed to be accessed starting at <b>Surface Min LOD</b>, which must be less than or equal to the number of MIP levels actually stored in memory for this surface. For sample* messages, the mip map access is clamped to be between the mipmap specified by the integer bits of the Min LOD and the ceiling of the value specified here. For ld* messages, out-of-bounds behavior results for LODs outside of the range specified in this field.
<b>For Render Target Surfaces:</b>
This field defines the MIP level that is currently being rendered into. This is the absolute MIP level on the surface and is not relative to the <b>Surface Min LOD</b> field, which is ignored for render target surfaces.
<b>For Other Surfaces:</b>
This field is reserved : MBZ
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>The <b>LOD</b> of a render target must be the same as the <b>LOD</b> of the other render target(s) and of the depth buffer (defined in 3DSTATE_DEPTH_BUFFER).
For render targets with YUV surface formats, the <b>LOD</b> must be zero.
For sampling engine surfaces with YCRCB* or PLANAR* surface format, <b>MIP Count</b> must be zero.</td></tr>
</table></td></tr>
<tr><td>6</td><td>31</td><td><b>Reserved</b>
<table>
<tr><td>Exists If:</td><td>([Surface Format] != 'PLANAR')</td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>31</td><td><b>Separate UV Plane Enable</b>
<table>
<tr><td>Exists If:</td><td>([Surface Format] == 'PLANAR')</td></tr>
<tr><td>Format:</td><td>Enable</td></tr>
</table>
If enabled, this field indicates that the U and V are present as separate planes. If disabled, the UV data is interleaved on a single plane.
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>See the section "Planar Memory Organization" for a description of how the size and location of the chroma planes (U and V) are calculated.</td></tr>
</table>
<table>
<tr><th>Workaround</th></tr>
<tr><td>The U and V planes, when separate (e.g. YV12) are treated as full-pitch (same width as Y plane). Zero padding in memory image will be required on each row of the U and V planes to sample</td></tr>
</table></td></tr>
</table>

<table>
<tr><th colspan="3">RENDER_SURFACE_STATE</th></tr>
<tr><td></td><td></td><td>correctly.</td></tr>
<tr><td></td><td>30</td><td><b>Reserved</b>
<table>
<tr><td>Exists If:</td><td>([Surface Format] == 'PLANAR')</td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>30:16</td><td><b>Auxiliary Surface QPitch</b>
<table>
<tr><td>Exists If:</td><td>([Surface Format] != 'PLANAR')</td></tr>
<tr><td>Format:</td><td>QPitch[16:2]</td></tr>
</table>
This field specifies the distance in rows between array slices on the auxiliary surface.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>[4h,1FFFCh]</td><td></td><td>in multiples of 4 (low 2 bits missing)</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>This field must be set to an integer multiple of the <b>Surface Vertical Alignment</b></td></tr>
<tr><td>Software must ensure that this field is set to a value sufficiently large such that the array slices in the auxiliary surface do not overlap. Refer to the Memory Data Formats section for information on how surfaces are stored in memory.</td></tr>
<tr><td>For non-multisampled render target's CCS auxiliary surface, QPitch must be computed with Horizontal Alignment = 128 and Surface Vertical Alignment = 256. These alignments are only for CCS buffer and not for associated render target.</td></tr>
</table></td></tr>
<tr><td></td><td>29:16</td><td><b>X Offset for U or UV Plane</b>
<table>
<tr><td>Exists If:</td><td>([Surface Format] == 'PLANAR')</td></tr>
<tr><td>Format:</td><td>U14</td></tr>
</table>
This field specifies the horizontal offset in pixels from the <b>Surface Base Address</b> to the start (origin) of the U plane or interleaved UV plane, depending on the setting of <b>Separate UV Plane Enable</b>.
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>This field must be a multiple of 4 (bits 1:0 MBZ).</td></tr>
<tr><td>If <b>Tiled Resource Mode</b> is enabled, this field must be a multiple of the tile width in pixels.</td></tr>
<tr><td><b>Auxiliary Surface Mode</b> is forced to AUX_NONE.</td></tr>
</table></td></tr>
<tr><td></td><td>15</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>14</td><td><b>Reserved</b>
<table>
<tr><td>Exists If:</td><td>([Surface Format] == 'PLANAR')</td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>14:12</td><td><b>Reserved</b>
<table>
<tr><td>Exists If:</td><td>([Surface Format] != 'PLANAR')</td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
</table>

RENDER_SURFACE_STATE

11:3

Auxiliary Surface Pitch

Exists If:	([Surface Format] != 'PLANAR')
Format:	U9-1 Pitch in #Tiles

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

Value	Name	Description
[0, 511]		-> [1 tile, 512 tiles]

13:0

Y Offset for U or UV Plane

Exists If:	([Surface Format] == 'PLANAR')
Format:	U14

This field specifies the vertical offset in rows from the **Surface Base Address** to the start (origin) of the U plane or interleaved UV plane, depending on the setting of **Separate UV Plane Enable**.

Programming Notes
For surfaces where **Surface Format** = PLANAR* and **Separate UV Plane** is Enabled, the Y Offset must be programmed in multiples of **half-rows**. For example, for a surface where Y is physically followed by U and then V in memory, the Y Offset to U plane would be (2*Y-Height). For all other PLANAR YUV formats this is programmed in multiples of full rows.
This field must be a multiple of 4 (bits 1:0 MBZ) for all YUV PLANAR surfaces.
If **Tiled Resource Mode** is enabled, this field must be a multiple of the tile height in rows.
Auxiliary Surface Mode is forced to AUX_NONE.

Workaround
Workaround : For formats PLANAR_420_* when this field is not a multiple of 4 the Out-of-Bounds Supression check must be disabled to avoid false out of bound detection.

2:0

Auxiliary Surface Mode

Exists If:	([Surface Format] != 'PLANAR')
Format:	U3

Specifies what type of surface the Auxiliary surface is. The Auxiliary surface has its own base address and pitch, but otherwise shares or overrides other fields set for the primary surface, detailed in the programming notes below.

Value	Name	Description
0h	AUX_NONE	No Auxiliary surface is used
1h	AUX_CCS_D	The Auxiliary surface is a CCS (Color Control Surface) with compression disabled or an MCS with compression enabled, depending on **Number of Multisamples**. MCS (Multisample Control Surface) is a special type of CCS.
2h	AUX_APPEND	The Auxiliary surface is an append buffer
3h	AUX_HIZ	The Auxiliary surface is a hierarchical depth buffer
4h	Reserved	

RENDER_SURFACE_STATE

5h	AUX_CCS_E	The Auxiliary surface is a CCS with compression enabled or an MCS with compression enabled, depending on **Number of Multisamples**.
6h-7h	Reserved	

Programming Notes
The CCS and hierarchical depth Auxiliary surface shares **Height, Width, Depth, Surface Type, Surface Array, Surface Min LOD, MIP Count / LOD, Surface Object Control State, Resource Min LOD**, and **Minimum Array Element** with the primary surface. The hierarchical depth Auxiliary surface uses **Surface Horizontal Alignment** of 16, **Surface Vertical Alignment** of 8, regardless of the primary surface's values for these fields. **X & Y Offset** are set to zero for the purpose of accessing the Auxiliary surface. If this field is set to AUX_HIZ, **Surface Format** must be be one of the following: R32_FLOAT, R24_UNORM_X8_TYPELESS, or R16_UNORM, and the format must match the format used when the surface was used as a depth buffer (with R channel corresponding to D channel).
CCS and hierarchical depth Auxiliary surfaces are TileY with **Tiled Resource Mode** of TRMODE_NONE regardless of the tile mode of the primary surface, and **Mip Tail Start LOD** is ignored for these surfaces.
The CCS Auxiliary surface for non-multisampled render targets has Horizontal Alignment = 128 and Vertical alignment = 64.
The CCS Auxiliary surface for **Number of Multisamples** > 1 uses **Surface Horizontal Alignment** of 16 and **Surface Vertical Alignment** of 4 regardless of the primary surface's values for these fields.
If this field is set to AUX_HIZ, **Number of Multisamples** must be MULTISAMPLECOUNT_1, and Surface Type cannot be SURFTYPE_3D.
If **Number of Multisamples** is MULTISAMPLECOUNT_1, AUX_CCS_E setting is only allowed if **Surface Format** is supported for Render Target Compression. This setting enables render target compression.
If **Number of Multisamples** is MULTISAMPLECOUNT_1, AUX_CCS_D setting is only allowed if **Surface Format** supported for Fast Clear. In addition, if the surface is bound to the sampling engine, **Surface Format** must be supported for Render Target Compression for surfaces bound to the sampling engine. For render target surfaces, this setting disables render target compression. For sampling engine surfaces, this mode behaves the same as AUX_CCS_E.
If **Number of Multisamples** is *not* MULTISAMPLECOUNT_1, both AUX_CCS_E and AUX_CCS_D settings indicate that the auxiliary surface is a multisample control surface (MCS), and multisample compression is enabled.
If **Number of Multisamples** is MULTISAMPLECOUNT_1, and if **Tiled Resource Mode** is NOT TRMODE_NONE, then, if CCS tile is NULL, Render Target Tiles represented by that CCS tile are assumed to be NULL by HW.

<table>
<tr><th colspan="3">RENDER_SURFACE_STATE</th></tr>
<tr><td>7</td><td>31</td><td><b>Memory Compression Mode</b>
Distinguishes Vertical from Horizontal compression.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>Horizontal <b>[Default]</b></td><td></td></tr>
<tr><td>1</td><td>Vertical</td><td></td></tr>
</table></td></tr>
<tr><td></td><td>30</td><td><b>Memory Compression Enable</b>
<table>
<tr><td>Format:</td><td>Enable</td></tr>
</table>
This surface may contain compressed or compressible pixels. Memory compression will be attempted for writes to this surface. Reads from this surface will check for compressed data.
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>The compression control must have 0 value for non-tileY modes. The Memory Compression Enable can be non-zero only for the surface state that has media messages. That is for 3d case the compression control bits will be 0 in normal surface state but can be non-zero in normal surface state for media messages. E.g. sample_unorm.</td></tr>
<tr><td>The only sampler messages supported with memory compression enabled are <i>sample_8x8</i>, <i>sample_unorm</i>, and SIMD16 <i>sample</i>.</td></tr>
<tr><td>Please refer to vol1a Memory Data Formats chapter section Media Memory Compression for more details, including format restrictions.</td></tr>
</table></td></tr>
<tr><td></td><td>29</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>28</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>27:25</td><td><b>Shader Channel Select Red</b>
<table>
<tr><td>Format:</td><td>Shader Channel Select Enumerated Type</td></tr>
</table>
Specifies which surface channel is read or written in the Red shader channel.
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>The Shader channel selects also define which shader channels are written to which surface channel. If the Shader channel select is SCS_ZERO or SCS_ONE then it is not written to the surface. If the shader channel select is SCS_RED it is written to the surface red channel and so on. If more than one shader channel select is set to the same surface channel only the first shader channel in RGBA order will be written. Each shader channel select must be set to the same surface channel (R = SCS_RED, G = SCS_GREEN, B = SCS_BLUE, A = SCS_ALPHA) if the surface is accessed via the sampler's sample_unorm* or sample_8x8 messages.</td></tr>
<tr><td>The Shader Channel Select fields do not affect the following sampling engine message types: resinfo, sampleinfo, LOD, and ld_mcs. These messages behave as if each Shader Channel Select is set to the same color surface channel.</td></tr>
<tr><td>For the sampling engine <i>gather4*</i> messages, the Gather4 Source Channel Select field in the message header defines which channel's Shader Channel Select is used to select the surface channel to be sampled. Other Shader Channel Select fields are ignored.</td></tr>
<tr><td>For the sampling engine <i>sample*_c</i> and <i>gather4*_c</i> messages, the compare operation always occurs on the red channel from the surface regardless of the setting of the Shader Channel Select fields.</td></tr>
</table></td></tr>
</table>

RENDER_SURFACE_STATE

		For Render Target, Red, Green and Blue Shader Channel Selects MUST be such that only valid components can be swapped i.e. only change the order of components in the pixel. Any other values for these Shader Channel Select fields are not valid for Render Targets. This also means that there MUST not be multiple shader channels mapped to the same RT channel. When multiple Channel selects have the same value and shader channel is disabled, disable channel writes 0s to memory. This behavior does not match with Data Port message via HDC. The output channel is undefined if the source is to a channel is not present for the current surface format. For example, If the surface format is R16_float and the shader channel select green specifies green as the source the output is undefined. It should instead select 0 which is the default for a missing color channel..
	24:22	**Shader Channel Select Green** Format: | Shader Channel Select Enumerated Type See **Shader Channel Select Red** for details.
	21:19	**Shader Channel Select Blue** Format: | Shader Channel Select Enumerated Type See **Shader Channel Select Red** for details.
	18:16	**Shader Channel Select Alpha** Format: | Shader Channel Select Enumerated Type See **Shader Channel Select Red** for details. **Programming Notes** For Render Target, this field MUST be programmed to value = SCS_ALPHA.
	15:12	**Reserved** Format: | MBZ
	11:0	**Resource Min LOD** Format: | U4.8 in LOD units **For Sampling Engine Surfaces:** This field indicates the most detailed LOD that is present in the resource underlying the surface. Refer to the "LOD Computation Pseudocode" section for the use of this field. **For Other Surfaces:** This field is ignored. **Value** | **Name** [0,14] | **Programming Notes** This field must be zero if the **Surface Format** is MONO8 This field must be zero if the **ChromaKey Enable** is enabled in the associated sampler.

<table>
<tr><th colspan="3">RENDER_SURFACE_STATE</th></tr>
<tr><td>8..9</td><td>63:0</td><td><b>Surface Base Address</b><table><tr><td>Format:</td><td>GraphicsAddress[63:0]SurfaceBase</td></tr></table>Specifies the byte-aligned base address of the surface.<table><tr><th>Programming Notes</th></tr><tr><td>• For SURFTYPE_BUFFER render targets, this field specifies the base address of first element of the surface. The surface is interpreted as a simple array of that single element type. The address must be naturally-aligned to the element size (e.g., a buffer containing R32G32B32A32_FLOAT elements must be 16-byte aligned).
• For SURFTYPE_BUFFER non-rendertarget surfaces, this field specifies the base address of the first element of the surface, computed in software by adding the surface base address to the byte offset of the element in the buffer. The base address must be aligned to element size.
• Linear depth buffer surface base addresses must be 64-byte aligned. Note that while render targets (color) can be SURFTYPE_BUFFER, depth buffers cannot.
• Mipmapped surfaces are stored in a "monolithic" (fixed) format, and only require a single address for the base MIP. All other MIPs are positioned relative to the base MIP.
• The Base Address for linear (non-tiled) render target surfaces and surfaces accessed with the typed surface read/write data port messages must be element-size aligned for Non-YUV surface formats, or a multiple of 2 element-sizes for YUV surface formats.
• Other linear (non-tiled) surfaces have no alignment requirements (byte alignment is sufficient).
• For tiled surfaces, the actual start of the surface can be offset from the Surface Base Address by the X Offset and Y Offset fields. Tiles are inherently page-aligned (4K or 64K).
• Certain message types used to access surfaces have more stringent alignment requirements. Please refer to the specific data-port message documentation for additional restrictions.</td></tr><tr><td>Tiled surface base addresses must be 4KB-aligned. Note that only the offsets from Surface Base Address are tiled, Surface Base Address itself is not transformed using the tiling algorithm.</td></tr><tr><td>Tiled surface base addresses must be tile aligned (64KB aligned for TileYS, 4KB aligned for all other tile modes). For 1D surfaces, the base address must be 64KB aligned if Tiled Resource Mode is TRMODE_64KB, and 4KB aligned if Tiled Resource Mode is TRMODE_4KB.
Compressed (BC*, ASTC, etc.) surface data is usually copied by re-describing each MIP/slice as a separate surface, using a size-equivalent RGBA format. But a MIP/slice within a packed <b>MIP Tail</b> doesn't have the tile-aligned <b>Surface Base Address</b> required for the re-description. This case must be specially handled by re-describing the packed <b>MIP Tail</b> as a single-MIP surface with the width/pitch/height/depth of a single tile, and then use drawing geometry to "reach out" to the desired tail slot (<i>x</i>, <i>y</i>, <i>z</i>) offset.</td></tr></table></td></tr>
</table>

RENDER_SURFACE_STATE

10..11	63:62	**Reserved** Exists If: ([Surface Format] == 'PLANAR') Format: MBZ
	61:48	**X Offset for V Plane** Exists If: ([Surface Format] == 'PLANAR') Format: U14 This field specifies the horizontal offset in pixels from the **Surface Base Address** to the start (origin) of the V plane.
	47:46	**Reserved** Exists If: ([Surface Format] == 'PLANAR') Format: MBZ
	45:32	**Y Offset for V Plane** Exists If: ([Surface Format] == 'PLANAR') Format: U14 This field specifies the vertical offset in rows from the **Surface Base Address** to the start (origin) of the V plane. **Programming Notes** For surfaces where **Surface Format** = PLANAR* and **Separate UV Plane** is Enabled, the Y Offset must be programmed in multiples of **half-rows**. For example, for a surface where Y is physically followed by U and then V in memory, the Y Offset to V plane would be (2*Y-Height+ U-Height). For all other PLANAR YUV formats this is programmed in multiples of full rows (e.g Y-Height + U-Height). This field must be a multiple of 4 (bits 1:0 MBZ). If **Tiled Resource Mode** is enabled, this field must be a multiple of the tile height in rows. This field is ignored if **Separate UV Plane Enable** is disabled.
	31:21	**Auxiliary Table Index for Media Compressed Surface** Exists If: [Memory Compression Enable] ==1 This field is valid only if Media Memory Compression is on for the surface(Memory Compression Enable == 1). In that case, the Auxiliary Surface Base address is never expected to be used and hence can be overloaded. This represents the 11 bit index into the table in memory which maps the surface to the auxiliary base address.
	63:12	**Auxiliary Surface Base Address** Exists If: ([Surface Format] != 'PLANAR') AND [Memory Compression Enable] == 0 Format: GraphicsAddress[63:12] Specifies the 4kbyte-aligned base address of the Auxiliary surface associated with the primary surface specified in other SURFACE_STATE fields.
	11	**Reserved** Format: MBZ

<table>
<tr><th colspan="3">RENDER_SURFACE_STATE</th></tr>
<tr><td></td><td>10</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table>
</td></tr>
<tr><td></td><td>9:5</td><td><b>Quilt Height</b>
<table><tr><td>Format:</td><td>U5</td></tr></table>
This field specifies the height of a quilted texture in units of quilt slices. Refer to the section on Quilted Textures for more details.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>[0,31]</td><td></td><td>representing height of quilt - 1 (y/v dimension)</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td><b>Programming Notes</b>
<ul>
<li>Only power-of-2 <b>Quilt Height</b> and <b>Quilt Width</b> values are allowed: (1,2,4,8,16,32) mapping to (0,1,3,7,15,31) values in the fields.</li>
<li>A surface is defined as a quilted texture if either <b>Quilt Height</b> or <b>Quilt Width</b> is nonzero (actual field value, not the incremented value).</li>
<li>A quilted texture
<ul>
<li>is only supported by the sampling engine (other shared functions will ignore the <b>Quilt Width</b> and <b>Quilt Height</b> field, behaving as if they are set to zero).</li>
<li>must have a <b>Surface Type</b> of SURFTYPE_2D.</li>
<li>must have <b>Number of Multisamples</b> set to NUMSAMPLES_1.</li>
<li>must have <b>Vertical Line Stride</b> set to 0.</li>
<li>must have <b>Auxiliary Surface Mode</b> set to AUX_NONE.</li>
<li><b>Depth</b> indicates the array dimension of the quilted texture if <b>Surface Array</b> is enabled. The valid range of <b>Depth</b> is [0, 2048 / (QuiltWidth * QuiltHeight) - 1], i.e. the total number of underlying array slices including quilt slices cannot exceed 2048.</li>
<li>cannot be accessed with any ld* message type or using a sampler with the <b>Non-Normalized Coordinate Enable</b> field enabled.</li>
</ul>
</li>
</ul>
</td></tr>
</table>
</td></tr>
<tr><td></td><td>4:0</td><td><b>Quilt Width</b>
<table><tr><td>Format:</td><td>U5</td></tr></table>
This field specifies the width of a quilted texture in units of quilt slices. Refer to the section on Quilted Textures for more details.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>[0,31]</td><td></td><td>representing width of quilt - 1 (x/u dimension)</td></tr>
</table>
</td></tr>
<tr><td>12</td><td>31:0</td><td><b>Reserved</b>
<table>
<tr><td>Exists If:</td><td>([Auxiliary Surface Mode] != 'AUX_CCS_D') AND ([Auxiliary Surface Mode] != 'AUX_CCS_E') AND ([Auxiliary Surface Mode] != 'AUX_HIZ')</td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
</table>
</td></tr>
</table>

RENDER_SURFACE_STATE

	31:0	**Red Clear Color** Exists If: [Auxiliary Surface Mode] == 'AUX_CCS_D' OR [Auxiliary Surface Mode] == 'AUX_CCS_E' Format: S31 (2's complement) for all SINT surface formats Format: U32 for all UINT surface formats Format: IEEE_FP for all other surface formats **For Sampling Engine Surfaces and Render Targets with Auxiliary Surface Mode set to AUX_CCS:** Specifies the clear value for the red channel. **For Other Surfaces:** This field is ignored. **Programming Notes** If Number of Multisamples is not MULTISAMPLECOUNT_1, only 0/1 values allowed If Number of Multisamples is MULTISAMPLECOUNT_1 AND if this RT is fast cleared with non-0/1 clear value, this RT must be partially resolved (refer to Partial Resolve operation) before binding this surface to Sampler.
	31:0	**Hierarchical Depth Clear Value** Exists If: [Auxiliary Surface Mode] == 'AUX_HIZ' Format: IEEE_Float If **Auxiliary Surface Mode** is AUX_HIZ, this field specifies the depth clear value associated with this surface. If disabled, this field is ignored.
13	31:0	**Green Clear Color** Exists If: [Auxiliary Surface Mode] == 'AUX_CCS_D' OR [Auxiliary Surface Mode] == 'AUX_CCS_E' Format: S31 (2's complement) for all SINT surface formats Format: U32 for all UINT surface formats Format: IEEE_FP for all other surface formats **For Sampling Engine Surfaces and Render Targets with Auxiliary Surface Mode set to AUX_CCS:** Specifies the clear value for the green channel. **For Other Surfaces:** This field is ignored. **Programming Notes** If programmed to non 0/1 values, SW must ensure a render target partial resolve pass before binding a cleared RT to texture.
	31:0	**Reserved** Exists If: [Auxiliary Surface Mode] == 'AUX_HIZ' Format: MBZ

RENDER_SURFACE_STATE

14	31:0	**Blue Clear Color** Exists If: [Auxiliary Surface Mode] == 'AUX_CCS_D' OR [Auxiliary Surface Mode] == 'AUX_CCS_E' Format: S31 (2's complement) for all SINT surface formats Format: U32 for all UINT surface formats Format: IEEE_FP for all other surface formats **For Sampling Engine Surfaces and Render Targets with Auxiliary Surface Mode set to AUX_CCS:** Specifies the clear value for the blue channel. **For Other Surfaces:** This field is ignored. **Programming Notes** If programmed to non 0/1 values, SW must ensure a render target partial resolve pass before binding a cleared RT to texture.
	31:0	**Reserved** Exists If: [Auxiliary Surface Mode] == 'AUX_HIZ' Format: MBZ
15	31:0	**Alpha Clear Color** Exists If: [Auxiliary Surface Mode] == 'AUX_CCS_D' OR [Auxiliary Surface Mode] == 'AUX_CCS_E' Format: S31 (2's complement) for all SINT surface formats Format: U32 for all UINT surface formats Format: IEEE_FP for all other surface formats **For Sampling Engine Surfaces and Render Targets with Auxiliary Surface Mode set to AUX_CCS:** Specifies the clear value for the alpha channel. **For Other Surfaces:** This field is ignored. **Programming Notes** If programmed to non 0/1 values, SW must ensure a render target partial resolve pass before binding a cleared RT to texture.
	31:0	**Reserved** Exists If: [Auxiliary Surface Mode] == 'AUX_HIZ' Format: MBZ

SAMPLER_STATE

<table>
<tr><th colspan="3">SAMPLER_STATE</th></tr>
<tr><td colspan="3">Source: BSpec
Exists If: //(MessageType != 'Deinterlace') && (MessageType != 'Sample_8x8')
Size (in bits): 128
Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000</td></tr>
<tr><td colspan="3">This is the normal sampler state used by all messages that use SAMPLER_STATE except sample_8x8 and deinterlace. The sampler state is stored as an array of up to 16 elements, each of which contains the dwords described here. The start of each element is spaced 4 dwords apart. The first element of the sampler state array is aligned to a 32-byte boundary.</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31</td><td><b>Sampler Disable</b>
<table><tr><td>Format:</td><td>Disable</td></tr></table>
This field allows the sampler to be disabled. If disabled, all output channels will return 0.</td></tr>
<tr><td></td><td>30</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>29</td><td><b>Texture Border Color Mode</b>
For some surface formats, the 32 bit border color is decoded differently based on the border color mode. In addition, the default value of channels not included in the surface may be affected by this field. Refer to the "Sampler Output Channel Mapping" table for the values of these channels, and for surface formats that may only support one of these modes. Also refer to the definition of SAMPLER_BORDER_COLOR_STATE for more details on the behavior of the two modes defined by this field.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>DX10/OGL</td><td>DX10/OGL mode for interpreting the border color</td></tr>
<tr><td>1h</td><td>DX9</td><td>DX9 and earlier mode for interpreting the border color</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>This feild must not be set to DX9 if there are null tiles in use</td></tr>
<tr><td>This field is required to be the same for every message over a period of time. A flush of the sampler cache must occur before a message with the opposite state of this field is delivered.</td></tr>
<tr><td>This field must be set to DX9 mode when used with surfaces that have Surface Format P4A4_UNORM or A4P4_UNORM.</td></tr>
<tr><td>This field must be set to DX10/OGL mode when used with surfaces that have Surface Format YCRCB_SWAPUV or YCRCB_SWAPY.</td></tr>
<tr><td>This field must be set to DX10/OGL mode if <b>Surface Format</b> for the associated surface is UINT OR SINT.</td></tr>
<tr><td>This field must be set to DX10/OGL mode if REDUCTION_MINIMUM or REDUCTION_MAXIMUM or message type is sample_min or sample_max.</td></tr>
</table></td></tr>
</table>

SAMPLER_STATE

28:27 **LOD PreClamp Mode**

This field determines whether the computed LOD is clamped to [max,min] mip level before the mag-vs-min determination is performed.

PRECLAMP_OGL: LOD pre-clamped to **Min LOD** and **Max LOD**

OpenGL API currently clamps LOD to the **Min LOD** and **Max LOD** (from Sampler State) prior to performing min/mag determination, and therefore it is expected that an OpenGL driver would need to set this field to PRECLAMP_OGL.

Value	Name	Description
0h	NONE	LOD PreClamp disabled
1h	Reserved	
2h	OGL	LOD PreClamp enabled (OGL mode)

26:22 **Coarse LOD Quality Mode**

Format:	U5

This field configures the coarse LOD image quality mode for the sample_d, sample_l, and sample_b messages in the sampling engine. In general, performance will increase and power consumption will decrease with each step of reduced quality (performance gain for sample_l and sample_b will be minimal).

Value	Name	Description
0h	Disabled	Full quality is enabled, matching prior products
01h-1Fh		Quality degrades with each larger value, performance improves with each larger value

Programming Notes
Although allowed, it is not recommended to program this field to a value greater than 17h to avoid masking the exponent which may generate incorrect LOD values.

SAMPLER_STATE

21:20 **Mip Mode Filter**

Format:	U2 Enumerated Type

This field determines if and how mip map levels are chosen and/or combined when texture filtering.

Value	Name	Description
0h	NONE	Disable mip mapping - force use of the mipmap level corresponding to Min LOD.
1h	NEAREST	Nearest, Select the nearest mip map
2h	Reserved	
3h	LINEAR	Linearly interpolate between nearest mip maps (combined with linear min/mag filters this is analogous to "Trilinear" filtering).

Programming Notes
MIPFILTER_LINEAR is not supported for surface formats that do not support "Sampling Engine Filtering" as indicated in the Surface Formats table unless using the sample_c message type or minimum/maximum operation.
Mip Mode Filter must be set to MIPFILTER_NONE or MIPFILTER_NEAREST if Surface Format for the associated surface is UINT or SINT. However, all settings of this field are allowed with UINT/SINT if a minimum or maximum operation is being performed.
Mip Mode Filter must be set to MIPFILTER_NONE for Planar YUV surfaces.

19:17 **Mag Mode Filter**

Format:	U3 Enumerated Type

This field determines how texels are sampled/filtered when a texture is being "magnified" (enlarged). For volume maps, this filter mode selection also applies to the 3rd (inter-layer) dimension.

Value	Name	Description
0h	NEAREST	Sample the nearest texel
1h	LINEAR	Bilinearly filter the 4 nearest texels
2h	ANISOTROPIC	Perform an "anisotropic" filter on the chosen mip level
4h-5h	Reserved	
6h	MONO	Perform a monochrome convolution filter
7h	Reserved	

Programming Notes
Only MAPFILTER_NEAREST and MAPFILTER_LINEAR are supported for surfaces of type SURFTYPE_3D.
Only MAPFILTER_NEAREST is supported for surface formats that do not support "Sampling Engine Filtering" as indicated in the Surface Formats table unless using the sample_c message type or minimum/maximum operation.

SAMPLER_STATE

MAPFILTER_MONO: Only CLAMP_BORDER texture addressing mode is supported. . Both Mag Mode Filter and Min Mode Filter must be programmed to MAPFILTER_MONO. Mip Mode Filter must be MIPFILTER_NONE. Only valid on surfaces with Surface Format MONO8 and with Surface Type SURFTYPE_2D.
MAPFILTER_ANISOTROPIC may cause artifacts at cube edges if enabled for cube maps with the TEXCOORDMODE_CUBE addressing mode.
MAPFILTER_ANISOTROPIC will be overridden to MAPFILTER_LINEAR when using a sample_l or sample_l_c message type or when Force LOD to Zero is set in the message header.
Both Mag Mode Filter and Min Mode Filter must be set to MAPFILTER_NEAREST if Surface Format for the associated surface is UINT or SINT. However, all settings of this field other than MAPFILTER_MONO are allowed with UINT/SINT if a minimum or maximum operation is being performed.

16:14 **Min Mode Filter**

Format:	U3 Enumerated Type

This field determines how texels are sampled/filtered when a texture is being "minified" (shrunk). For volume maps, this filter mode selection also applies to the 3rd (inter-layer) dimension.See Mag Mode Filter

Value	Name	Description
0h	NEAREST	Sample the nearest texel
1h	LINEAR	Bilinearly filter the 4 nearest texels
2h	ANISOTROPIC	Perform an "anisotropic" filter on the chosen mip level
4h-5h	Reserved	
6h	MONO	Perform a monochrome convolution filter
7h	Reserved	

13:1 **Texture LOD Bias**

Format:	S4.8 2's complement

Range: [-16.0, 16.0)
This field specifies the signed bias value added to the calculated texture map LOD prior to min-vs-mag determination and mip-level clamping. Assuming mipmapping is enabled, a positive LOD bias will result in a somewhat blurrier image (using less-detailed mip levels) and possibly higher performance, while a negative bias will result in a somewhat crisper image (using more-detailed mip levels) and may lower performance.

Programming Notes
There is no requirement or need to offset the LOD Bias in order to produce a correct LOD for texture filtering (as was required for correct bilinear and anisotropic filtering in some legacy devices).

<table>
<tr><th colspan="3">SAMPLER_STATE</th></tr>
<tr><td></td><td>0</td><td><b>LOD algorithm</b>
<table><tr><td>Format:</td><td>U1 Enumerated Type</td></tr></table>
Controls which algorithm is used for LOD calculation. Generally, the EWA approximation algorithm results in higher image quality than the legacy algorithm.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>LEGACY</td><td>Use the legacy algorithm for non-anisotropic filtering</td></tr>
<tr><td>1h</td><td>EWA Approximation</td><td>Use the new EWA approximation algorithm for anisotropic filtering</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>The EWA Algorithm should only be enabled for Anisotropic Filtering modes. It must not be enabled for non-anisotropic filtering as the increased accuracy of the LOD calculation will is not required and will increase the power and reduce overall efficiency.</td></tr>
</table>
</td></tr>
<tr><td>1</td><td>31:20</td><td><b>Min LOD</b>
<table><tr><td>Format:</td><td>U4.8 in LOD units</td></tr></table>
<table>
<tr><td>Range: [0.0, 14.0], where the upper limit is also bounded by the Max LOD.</td></tr>
<tr><td>This field specifies the minimum value used to clamp the computed LOD after LOD bias is applied. Note that the minification-vs.-magnification status is determined after LOD bias and before this maximum (resolution) mip clamping is applied. The integer bits of this field are used to control the "maximum" (highest resolution) mipmap level that may be accessed (where LOD 0 is the highest resolution map). The fractional bits of this value effectively clamp the inter-level trilinear blend factor when trilinear filtering is in use.</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>If Min LOD is greater than Max LOD, Min LOD takes precedence, i.e. the resulting LOD will always be Min LOD.</td></tr>
<tr><td>This field must be zero if the Min or Mag Mode Filter is set to MAPFILTER_MONO</td></tr>
</table>
</td></tr>
<tr><td></td><td>19:8</td><td><b>Max LOD</b>
<table><tr><td>Format:</td><td>U4.8 in LOD units</td></tr></table>
<table>
<tr><td>Range: [0.0, 14.0]</td></tr>
<tr><td>This field specifies the maximum value used to clamp the computed LOD after LOD bias is applied. Note that the minification-vs.-magnification status is determined after LOD bias and before this minimum (resolution) mip clamping is applied.The integer bits of this field are used to control the "minimum" (lowest resolution) mipmap level that may be accessed.The fractional bits of this value effectively clamp the inter-level trilinear blend factor when trilinear filtering is in use.Force the mip map access to be between the mipmap specified by the integer bits of the Min LOD and the ceiling of the value specified here.</td></tr>
</table>
</td></tr>
</table>

SAMPLER_STATE

7 **ChromaKey Enable**

Format:	Enable This field enables the chroma key function.

Programming Notes
Supported only on a specific subset of surface formats. See section titled: "Surface Formats" in thsi volume for supported formats. This field must be disabled if min or mag filter is MAPFILTER_MONO or MAPFILTER_ANISOTROPIC. This field must be disabled if used with a surface of type SURFTYPE_3D.

6:5 **ChromaKey Index**

Format:	U2

Range: [0, 3]
This field specifies the index of the ChromaKey Table entry associated with this Sampler. This field is a "don't care" unless **ChromaKey Enable** is ENABLED.

4 **ChromaKey Mode**

Format:	U1 Enumerated Type

This field specifies the behavior of the device in the event of a ChromaKey match. This field is ignored if ChromaKey is disabled.
KEYFILTER_REPLACE_BLACK: In this mode, each texel that matches the chroma key is replaced with (0,0,0,0) (black with alpha=0) prior to filtering. For YCrCb surface formats, the black value is A=0, R(Cr)=0x80, G(Y)=0x10, B(Cb)=0x80. This will tend to darken/fade edges of keyed regions. Note that the pixel pipeline must be programmed to use the resulting filtered texel value to gain the intended effect, e.g., handle the case of a totally keyed-out region (filtered texel alpha==0) through use of alpha test, etc.

Value	Name	Description
0h	KEYFILTER_KILL_ON_ANY_MATCH	In this mode, if any contributing texel matches the chroma key, the corresponding pixel mask bit for that pixel is cleared. The result of this operation is observable only if the Killed Pixel Mask Return flag is set on the input message.
1h	KEYFILTER_REPLACE_BLACK	In this mode, each texel that matches the chroma key is replaced with (0,0,0,0) (black with alpha=0) prior to filtering. For YCrCb surface formats, the black value is A=0, R(Cr)=0x80, G(Y)=0x10, B(Cb)=0x80. This will tend to darken/fade edges of keyed regions. Note that the pixel pipeline must be programmed to use the resulting filtered texel value to gain the intended effect, e.g., handle the case of a totally keyed-out

<table>
<tr><th colspan="3">SAMPLER_STATE</th></tr>
<tr><td></td><td></td><td><table><tr><td></td><td></td><td>region (filtered texel alpha==0) through use of alpha test, etc.</td></tr></table></td></tr>
<tr><td></td><td>3:1</td><td><b>Shadow Function</b>
<table><tr><td>Format:</td><td>U3 Enumerated Type</td></tr></table>
This field is used for shadow mapping support via the sample_c message type, and specifies the specific comparison operation to be used. The comparison is between the texture sample red channel (except for alpha-only formats which use the alpha channel), and the "ref" value provided in the input message.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0h</td><td>PREFILTEROP ALWAYS</td></tr>
<tr><td>1h</td><td>PREFILTEROP NEVER</td></tr>
<tr><td>2h</td><td>PREFILTEROP LESS</td></tr>
<tr><td>3h</td><td>PREFILTEROP EQUAL</td></tr>
<tr><td>4h</td><td>PREFILTEROP LEQUAL</td></tr>
<tr><td>5h</td><td>PREFILTEROP GREATER</td></tr>
<tr><td>6h</td><td>PREFILTEROP NOTEQUAL</td></tr>
<tr><td>7h</td><td>PREFILTEROP GEQUAL</td></tr>
</table></td></tr>
<tr><td></td><td>0</td><td><b>Cube Surface Control Mode</b>
<table><tr><td>Format:</td><td>U1 Enumerated Type</td></tr></table>
When sampling from a SURFTYPE_CUBE surface, this field controls whether the TC* Address Control Mode fields are interpreted as programmed or overridden to TEXCOORDMODE_CUBE.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0h</td><td>PROGRAMMED</td></tr>
<tr><td>1h</td><td>OVERRIDE</td></tr>
</table></td></tr>
<tr><td>2</td><td>31:30</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>29:28</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>27:26</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>31:24</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>25:24</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>23:6</td><td><b>Indirect State Pointer</b>
<table>
<tr><th>Description</th></tr>
<tr><td>This pointer is relative to the Dynamic State Base Address.</td></tr>
</table></td></tr>
<tr><td></td><td>5</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>4</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>3</td><td><b>Reserved</b></td></tr>
</table>

<table>
<tr><th colspan="3">SAMPLER_STATE</th></tr>
<tr><td></td><td>2</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>1</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>0</td><td><b>LOD Clamp Magnification Mode</b>
<table>
<tr><td>Format:</td><td>U1 Enumerated Type</td></tr>
</table>
This field allows the flexibility to control how LOD clamping is handled when in magnification mode.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>MIPNONE</td><td>When in magnification mode, Sampler will clamp LOD as if the <b>Mip Mode Filter</b> is MIPFILTER_NONE. This is how OpenGL defines magnification, and therefore it is expected that those drivers would not set this bit.</td></tr>
<tr><td>1h</td><td>MIPFILTER</td><td>When in magnification mode, Sampler will clamp LOD based on the value of <b>Mip Mode Filter</b>.</td></tr>
</table>
</td></tr>
<tr><td>3</td><td>31:24</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>25:24</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table>
</td></tr>
<tr><td></td><td>23:22</td><td><b>Reduction Type</b>
<table>
<tr><td>Format:</td><td>U2 Enumerated Type</td></tr>
</table>
This field defines the type of reduction that will be performed on the texels in the footprint defined by the <b>Min/Mag/Mip Filter Mode</b> fields. This field is ignored if <b>Reduction Type Enable</b> is disabled.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>STD_FILTER</td><td>standard filter</td></tr>
<tr><td>1h</td><td>COMPARISON</td><td>comparison followed by standard filter</td></tr>
<tr><td>2h</td><td>MINIMUM</td><td>minimum of footprint</td></tr>
<tr><td>3h</td><td>MAXIMUM</td><td>maximum of footprint</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>The following message types ignore this field: <i>sample_min, sample_max, sample_unorm*, resinfo, sampleinfo, LOD, ld*, sample_8x8</i>.</td></tr>
<tr><td>If the current min/mag filter mode is MAPFILTER_MONO, this field is ignored.</td></tr>
<tr><td>The <i>sample_c, sample_l_c, sample_d_c, sample_b_c, gather4_c</i>, and <i>gather4_po_c</i> message types, when used with STD_FILTER, MINIMUM, or MAXIMUM settings of this field, perform the operation of the message of the same name without the "_c". The ref parameter is ignored by hardware.</td></tr>
<tr><td>For message types not listed above, when used with COMPARISON setting of this field, perfom the operation of the message of the same name with "_c" included. The ref parameter used by the operation (since it is not delivered in the message) is set to zero.</td></tr>
</table>
</td></tr>
</table>

SAMPLER_STATE

Restrictions applying to the message whose behavior is being performed must be followed. For example, a sample message used with COMPARISON reduction filter must follow all of the restrictions of *sample_c*. An exception to this is the MINIMUM and MAXIMUM reduction types allow SURFTYPE_1D, 2D, 3D, and CUBE, including with **Surface Array** enabled, even though the sample_min/max messages only allow 2D.

Restrictions applying to the message delivered need not be followed. For example, a *sample_c* message used with STD_FILTER reduction filter needs to follow only the restrictions of sample, not the restrictions of *sample_c*.

21:19 **Maximum Anisotropy**

Format:	U3 Enumerated Type

This field clamps the maximum value of the anisotropy ratio used by the MAPFILTER_ANISOTROPIC filter (Min or Mag Mode Filter).

Value	Name	Description
0h	RATIO 2:1	At most a 2:1 aspect ratio filter is used
1h	RATIO 4:1	At most a 4:1 aspect ratio filter is used
2h	RATIO 6:1	At most a 6:1 aspect ratio filter is used
3h	RATIO 8:1	At most a 8:1 aspect ratio filter is used
4h	RATIO 10:1	At most a 10:1 aspect ratio filter is used
5h	RATIO 12:1	At most a 12:1 aspect ratio filter is used
6h	RATIO 14:1	At most a 14:1 aspect ratio filter is used
7h	RATIO 16:1	At most a 16:1 aspect ratio filter is used

18 **U Address Mag Filter Rounding Enable**

Format:	Enable

Controls whether the texture address is rounded or truncated before being used to select texels to sample. Provides independent control of rounding on one texture address dimension (U/V/R) in either mag or min filter mode.

Programming Notes
Hardware will **not** force rounding enable.

17 **U Address Min Filter Rounding Enable**

Format:	Enable

Controls whether the texture address is rounded or truncated before being used to select texels to sample. Provides independent control of rounding on one texture address dimension (U/V/R) in either mag or min filter mode.

Programming Notes
Hardware will **not** force rounding enable.

16 **V Address Mag Filter Rounding Enable**

Format:	Enable

Controls whether the texture address is rounded or truncated before being used to select texels to sample. Provides independent control of rounding on one texture address dimension (U/V/R)

SAMPLER_STATE

<table>
<tr><td></td><td></td><td>in either mag or min filter mode.<table><tr><th>Programming Notes</th></tr><tr><td>Hardware will <b>not</b> force rounding enable.</td></tr></table></td></tr>
<tr><td></td><td>15</td><td><b>V Address Min Filter Rounding Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>Controls whether the texture address is rounded or truncated before being used to select texels to sample. Provides independent control of rounding on one texture address dimension (U/V/R) in either mag or min filter mode.<table><tr><th>Programming Notes</th></tr><tr><td>Hardware will <b>not</b> force rounding enable.</td></tr></table></td></tr>
<tr><td></td><td>14</td><td><b>R Address Mag Filter Rounding Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>Controls whether the texture address is rounded or truncated before being used to select texels to sample. Provides independent control of rounding on one texture address dimension (U/V/R) in either mag or min filter mode.<table><tr><th>Programming Notes</th></tr><tr><td>Hardware will <b>not</b> force rounding enable.</td></tr></table></td></tr>
<tr><td></td><td>13</td><td><b>R Address Min Filter Rounding Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>Controls whether the texture address is rounded or truncated before being used to select texels to sample. Provides independent control of rounding on one texture address dimension (U/V/R) in either mag or min filter mode.<table><tr><th>Programming Notes</th></tr><tr><td>Hardware will <b>not</b> force rounding enable.</td></tr></table></td></tr>
<tr><td></td><td>12:11</td><td><b>Trilinear Filter Quality</b><table><tr><td>Format:</td><td>U2 Enumerated Type</td></tr></table>Selects the quality level for the trilinear filter.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td>FULL</td><td>Full Quality. Both mip maps are sampled under all circumstances.</td></tr><tr><td>1</td><td>TRIQUAL_HIGH/MAG_CLAMP_MIPFILTER</td><td>If High Quality. Same as full quality.</td></tr><tr><td>2</td><td>MED</td><td>If Medium Quality. If the contribution of one mip map is less than 25%, only the other mip map contributes.</td></tr><tr><td>3</td><td>LOW</td><td>If Low Quality. If the contribution of one mip map is less than 37.5%, only the other mip map contributes.</td></tr></table></td></tr>
</table>

<table>
<tr><th colspan="3">SAMPLER_STATE</th></tr>
<tr><td></td><td>10</td><td><b>Non-normalized Coordinate Enable</b>
<table><tr><td>Format:</td><td>Enable</td></tr></table>
This field, if enabled, specifies that the input coordinates (U/V/R) are in non-normalized space, where each integer increment is one texel on LOD 0. If disabled, coordinates are normalized, where the range 0 to 1 spans the entire surface.
<table><tr><th>Programming Notes</th></tr>
<tr><td>The following state must be set as indicated if this field is <i>enabled</i>:
<ul>
<li>TCX/Y/Z Address Control Mode must be TEXCOORDMODE_CLAMP, TEXCOORDMODE_HALF_BORDER, or TEXCOORDMODE_CLAMP_BORDER.</li>
<li>Surface Type must be SURFTYPE_2D or SURFTYPE_3D.</li>
<li>Mag Mode Filter must be MAPFILTER_NEAREST or MAPFILTER_LINEAR.</li>
<li>Min Mode Filter must be MAPFILTER_NEAREST or MAPFILTER_LINEAR.</li>
<li>Mip Mode Filter must be MIPFILTER_NONE.</li>
<li>Min LOD must be 0.</li>
<li>Max LOD must be 0.</li>
<li>MIP Count must be 0.</li>
<li>Surface Min LOD must be 0.</li>
<li>Texture LOD Bias must be 0.</li>
</ul></td></tr></table></td></tr>
<tr><td></td><td>9</td><td><b>Reduction Type Enable</b>
<table><tr><td>Format:</td><td>Enable</td></tr></table>
This field enables the <b>Reduction Type</b> field to modify the behavior of messages based on its setting. If this field is disabled, all messages behave as defined and the <b>Reduction Type</b> field is ignored.</td></tr>
<tr><td></td><td>8:6</td><td><b>TCX Address Control Mode</b>
<table><tr><td>Format:</td><td>Texture Coordinate Mode Enumerated Type</td></tr></table>
Controls how the 1st (TCX, aka U) component of input texture coordinates are mapped to texture map addresses - specifically, how coordinates "outside" the texture are handled (wrap/clamp/mirror). The setting of this field is subject to being overridden by the Cube Surface Control Mode field when sampling from a SURFTYPE_CUBE surface.
<table><tr><th>Programming Notes</th></tr>
<tr><td>When using cube map texture coordinates, each TC component must have the same Address Control Mode.</td></tr>
<tr><td>When TEXCOORDMODE_CUBE is not used accessing a cube map, the map's Cube Face Enable field must be programmed to 111111b (all faces enabled).</td></tr>
<tr><td>MAPFILTER_MONO: Texture addressing modes must all be set to TEXCOORDMODE_CLAMP_BORDER. The <b>Border Color</b> is ignored in this mode, a constant value of 0 is used for border color. Software must pad the border texels within the map itself with 0.</td></tr>
<tr><td>If <b>Surface Format</b> is PLANAR*, this field must be set to TEXCOORDMODE_CLAMP.</td></tr>
</table></td></tr>
<tr><td></td><td>5:3</td><td><b>TCY Address Control Mode</b></td></tr>
</table>

SAMPLER_STATE

		Format: Texture Coordinate Mode Enumerated Type Controls how the 2nd (TCY, aka V) component of input texture coordinates are mapped to texture map addresses - specifically, how coordinates "outside" the texture are handled (wrap/clamp/mirror). See Address TCX Control Mode above for details **Programming Notes** If this field is set to TEXCOORDMODE_CLAMP_BORDER or TEXCOORDMODE_HALF_BORDER and a 1D surface is sampled, incorrect blending with the border color in the vertical direction may occur.
	2:0	**TCZ Address Control Mode** Format: Texture Coordinate Mode Enumerated Type Controls how the 3rd (TCZ) component of input texture coordinates are mapped to texture map addresses - specifically, how coordinates "outside" the texture are handled (wrap/clamp/mirror).See Address TCX Control Mode above for details

SubslicePool

SubslicePool

Source:	BSpec
Size (in bits):	32
Default Value:	0x00000000

These fields specify the assignment of EUs within the given subslice to one of two pools. Bit 7..0 correspond to EU7..0 in the subslice. Assignment of the corresponding EU is to Pool-0 if its bit=0, or Pool-1, if its bit=1. Bit values in bit positions of non-existent or disabled EUs must be set to 0.

DWord	Bit	Description
0	31:24	**Subslice3** Default Value: 00h Default Value Format: U8
	23:16	**Subslice2** Default Value: 00h Default Value Format: U8
	15:8	**Subslice1** Default Value: 00h Default Value Format: U8
	7:0	**Subslice0** Default Value: 00h Default Value Format: U8

Surface Binding Table Index Message Descriptor Control Field

MDC_BTS - Surface Binding Table Index Message Descriptor Control Field

Source:	BSpec
Size (in bits):	8
Default Value:	0x00000000

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>7:0</td><td><b>Binding Table Index</b>
<table>
<tr><td>Format:</td><td>Enumeration</td></tr>
</table>
Specifies the Binding Table index for the message, which must be a Surface State Model.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>00h-0EFh</td><td>BTS</td><td>Index of Binding Table State Surfaces</td></tr>
<tr><td>F0h-0FBh</td><td>Reserved</td><td>Reserved for future use</td></tr>
<tr><td>0FCh</td><td>Reserved</td><td>Reserved for future use</td></tr>
<tr><td>Others</td><td>Reserved</td><td>Ignored</td></tr>
</table>
</td></tr>
</table>

Surface or Stateless Binding Table Index Message Descriptor Control Field

MDC_BTS_A32 - Surface or Stateless Binding Table Index Message Descriptor Control Field

Source:	BSpec
Size (in bits):	8
Default Value:	0x00000000

DWord	Bit	Description
0	7:0	**Binding Table Index**

Format:	Enumeration

Specifies the surface for the message, either Surface State Model or Stateless.

Value	Name	Description
00h-0EFh	BTS	Index of Binding Table State Surfaces
F0h-0FBh	Reserved	Reserved for future use
0FCh	Reserved	Reserved for future use
0FFh	A32_A64	Specifies a A32 or A64 Stateless access that is locally coherent (coherent within a thread group)
0FDh	A32_A64_NC	Specifies a A32 or A64 Stateless access that is non-coherent (coherent within a thread).
Others	Reserved	Ignored

Restriction
Restriction : When using A32_A64_NC, SW must ensure that 2 threads do not both access the same cache line (64B)

SW Generated BINDING_TABLE_STATE

SW Generated BINDING_TABLE_STATE

Source:	BSpec
Size (in bits):	32
Default Value:	0x00000000

Description
The binding table binds surfaces to logical resource indices used by shaders and other compute engine kernels. It is stored as an array of up to 256 elements, each of which contains one dword as defined here. The start of each element is spaced one dword apart.
The first element of the binding table is aligned to a 64-byte boundary.
Binding table indexes beyond 256 will automatically be mapped to entry 0 by the HW, w/ the exception of any messages which support the special indexes 255 and 254.

DWord	Bit	Description
0	31:6	**Surface State Pointer** Format: SurfaceStateOffset[31:6] This 64-byte aligned address points to a surface state block. This pointer is relative to the **Surface State Base Address**
	5	**Reserved** Format: MBZ
	4:0	**Reserved** Format: MBZ