Intel® OpenSource HD Graphics Programmer's Reference Manual (PRM)

Volume 1 Part 5: Graphics Core™ – Video Codec Engine Command Streamer (Ivy Bridge)

For the 2012 Intel® Core™ Processor Family

May 2012

Revision 1.0

NOTICE:
This document contains information on products in the design phase of development, and Intel reserves the right to add or remove product features at any time, with or without changes to this open source documentation.

Creative Commons License

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.

Contents

1. Video Codec Engine Command Streamer

Full decode pipeline as well as encode pipeline are implemented in VCE.

VCE has its own command streamer and operates completely independently of the render (3D/Media) pipeline command streamer.

1.1 Registers for Video Codec

1.1.1 Introduction

This command streamer supports a completely independent set of registers. Only a subset of the MI Registers is supported for this 2nd command streamer. The effort is to keep the registers at the same offset as the render command streamer registers. The base of the registers for the video decode engine will be defined per project, the offsets will be maintained.

Project	Base Address Value for the memory interface register offset for the Bit Stream Command Stream
	0x10000 eg: The Ring buffer tail pointer will be 0x10000 + 0x2030

1.1.2 Virtual Memory Control

MFX engine Supports a 2-level mapping scheme for PPGTT, consisting of a first-level page directory containing page table base addresses, and the page tables themselves on the 2nd level, consisting of page addresses.

1.1.2.1 VCS_PP_DCLV – VCS PPGTT Directory Cacheline Valid Register

VCS_PP_DCLV - VCS PPGTT Directory Cacheline Valid Register	
Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000, 0x00000000
Access:	R/W
Size (in bits):	64
Address:	12220h

This register controls update of the on-chip PPGTT Directory Cache during a context restore. Bits that are set will trigger the load of the corresponding 16 directory entry group.

This register is restored with context (prior to restoring the on-chip directory cache itself). This register is also restored when switching to a context whose LRCA matches the current CCID if the **Force PD Restore** bit is set in the context descriptor. The context image of this register must be updated and maintained by SW; SW should not

VCS_PP_DCLV - VCS PPGTT Directory Cacheline Valid Register

normally need to read this register.

This register can also effectively be used to limit the size of a processes' virtual address space. Any access by a process that requires a PD entry in a set that is not enabled in this register will cause a fatal error, and no fetch of the PD entry will be attempted.

DWord	Bit	Description
0	63:32	**Reserved** Format: MBZ
	31:0	**PPGTT Directory Cache Restore [1..32] 16 entries** Format: Enable[32] If set, the [1st..32nd] 16 entries of the directory cache are considered valid and will be brought in on context restore. If clear, these entries are considered invalid and fetch of these entries will not be attempted.

1.1.2.2 VCS_EXCC—Execute Condition Code Register

VCS_EXCC - VCS Execute Condition Code Register

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	R/W,RO
Size (in bits):	32
Trusted Type:	1
Address:	12028h

This register contains user defined and hardware generated conditions that are used by MI_WAIT_FOR_EVENT commands. An MI_WAIT_FOR_EVENT instruction excludes the executing ring from arbitration if the selected event evaluates to a 1, while instruction is discarded if the condition evaluates to a 0. Once excluded, a ring is enabled into arbitration when the selected condition evaluates to a 0.

This register also contains control for the invalidation of indirect state pointers on context restore.

DWord	Bit	Description
0	31:16	**Mask Bits** Format: Mask[15:0] These bits serve as a write enable for bits 15:0. If this register is written with any of these bits clear the corresponding bit in the field 15:0 will not be modified. Reading these bits always returns 0s.
	15:8	**Reserved** Format: MBZ
	4:0	**User Defined Condition Codes** The software may signal a Stream Semaphore by setting the Mask bit and Signal Bit together to match the bit field specified in a WAIT_FOR_EVENT (Semaphore).

1.1.2.2.1 VCS_HWS_PGA — VCS Hardware Status Page Address Register

VCS_HWS_PGA - VCS Hardware Status Page Address Register

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Address:	04180h

This register is used to program the 4 KB-aligned System Memory address of the Hardware Status Page used to report hardware status into (typically cacheable) System Memory.

Programming Notes	Project
If this register is written, a workload must subsequently be dispatched to the video command streamer.	

DWord	Bit	Description
0	31:12	**Address** Format: GraphicsAddress[31:12] This field is used by SW to specify Bits 31:12 of the 4 KB-aligned System Memory address of the 4 KB page known as the Hardware Status Page. The Global GTT is used to map this page from the graphics virtual address to physical address. **Programming Notes** If the **Per-Process Virtual Address Space** bit is set, HW requires that the status page is programmed to allow for the context switch status to be reported.
	11:1	**Reserved** Format: MBZ
	0	**Translation In Progress** Format: U1 This field indicates that the translation for the hardware status page from the graphics virtual address to the physical address is pending. Software can use this indicator to prevent updating the status page when there is a pending cycle for translation.

1.1.3 Mode and Misc Ctrl Registers

1.1.3.1 2nd Level Batch Buffer Address

2nd Level Batch Buffer Address

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	12144h

This register is to read the current value of the 2nd level batch buffer address. Since the 2nd level batch buffer logic is shared with the C6 work-around batch buffer, this also shows the work-around address when it is active.

DWord	Bit	Description
0	31:2	**WA Batch Buffer Address** Format: U30 Pointer to the WA Batch Buffer Address.
	1:0	**Reserved** Format: MBZ

1.1.3.2 VCS_CXT_SIZE - VCS Context Sizes

<table>
<tr><th colspan="2">VCS_CXT_SIZE - VCS Context Sizes</th></tr>
<tr><td>Register Space:</td><td>MMIO: 0/2/0</td></tr>
<tr><td>Source:</td><td>VideoCS</td></tr>
<tr><td>Default Value:</td><td>0x00040D00</td></tr>
<tr><td>Access:</td><td>Read/32 bit Write Only</td></tr>
<tr><td>Size (in bits):</td><td>32</td></tr>
<tr><td>Address:</td><td>121A8h</td></tr>
</table>

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:21</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>20:16</td><td><b>VCS Context Size</b><table><tr><td>Format:</td><td>U5</td></tr></table><table><tr><th>Value</th><th>Name</th><th>Project</th></tr><tr><td>4h</td><td><b>[Default]</b></td><td></td></tr></table></td></tr>
<tr><td></td><td>15:13</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>12:8</td><td><b>VCR Context Size</b><table><tr><td>Format:</td><td>U3</td></tr></table><table><tr><th>Value</th><th>Name</th><th>Project</th></tr><tr><td>Dh</td><td><b>[Default]</b></td><td></td></tr></table></td></tr>
<tr><td></td><td>7:5</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
</table>

1.1.3.3 VCS_MI_MODE — VCS Mode Register for Software Interface

VCS_MI_MODE - VCS Mode Register for Software Interface

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Address:	1209Ch-1209Fh

The MI_MODE register contains information that controls software interface aspects of the command parser.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:16</td><td><b>Masks</b>
A 1 in a bit in this field allows the modification of the corresponding bit in Bits 15:0.</td></tr>
<tr><td></td><td>15</td><td><b>Suspend Flush</b>
<table><tr><td>Mask:</td><td>MMIO(0x209c)#31</td></tr></table>
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>No Delay</td><td>HW will not delay flush, this bit will get cleared by MI_SUSPEND_FLUSH as well</td></tr>
<tr><td>1h</td><td>DelayFlush</td><td>Suspend flush is active</td></tr>
</table></td></tr>
<tr><td></td><td>14:12</td><td><b>Reserved</b>
<table><tr><td>Access:</td><td>R/W</td></tr></table></td></tr>
<tr><td></td><td>11</td><td><b>Invalidate UHPTR enable</b>
If bit set H/W clears the valid bit of BCS_UHPTR (4134h, bit 0) when current active head pointer is equal to UHPTR.</td></tr>
<tr><td></td><td>10</td><td><b>Reserved</b>
<table><tr><td></td><td></td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>9</td><td><b>Ring Idle (Read Only Status bit)</b>
<table><tr><td>Access:</td><td>RO</td></tr></table>
<i>Writes to this bit are not allowed.</i>
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0</td><td>Parser not idle</td></tr>
<tr><td>1</td><td>Parser idle</td></tr>
</table></td></tr>
<tr><td></td><td>8</td><td><b>Stop Ring</b>
Software must set this bit to force the Ring and Command Parser to Idle. Software must read a “1” in Ring Idle bit after setting this bit to ensure that the hardware is idle.
<i>Software must clear this bit for Ring to resume normal operation.</i>
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0</td><td>Normal Operation</td></tr>
<tr><td>1</td><td>Parser is turned off</td></tr>
</table></td></tr>
<tr><td></td><td>7:0</td><td><b>Reserved</b>
<table><tr><td>Access:</td><td>R/W</td></tr></table></td></tr>
</table>

1.1.3.4 MFX_MODE – Video Mode Register

MFX_MODE - Video Mode Register	
Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	1229Ch

This register contains a control bit for the 2-level PPGTT functions.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:16</td><td><b>Mask Bits</b><table><tr><td>Format:</td><td>Mask[15:0]</td></tr></table>Must be set to modify corresponding bit in Bits 15:0. (All implemented bits)</td></tr>
<tr><td></td><td>14</td><td><b>Reserved</b><table><tr><td></td><td></td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>13:10</td><td><b>Reserved</b><table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>9</td><td><b>Per-Process GTT Enable</b><table><tr><td>Format:</td><td>Enable Per-Process GTT BS Mode Enable</td></tr></table><table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>PPGTT Disable <b>[Default]</b></td><td>When clear, the Global GTT will be used to translate memory access from designated commands and for commands that select the PPGTT as their translation space.</td></tr><tr><td>1h</td><td>PPGTT Enable</td><td>When set, the PPGTT will be used to translate memory access from designated commands and for commands that select the PPGTT as their translation space.</td></tr></table></td></tr>
<tr><td></td><td>7</td><td><b>Reserved</b><table><tr><td></td><td></td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>6:5</td><td><b>Reserved</b><table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>4:0</td><td><b>Reserved</b><table><tr><td></td><td></td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
</table>

1.1.3.5 VCS_INSTPM—VCS Instruction Parser Mode Register

VCS_INSTPM - VCS Instruction Parser Mode Register

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Address:	120C0h-120C3h

The VCS_INSTPM register is used to control the operation of the VCS Instruction Parser. Certain classes of instructions can be disabled (ignored) – often useful for detecting performance bottlenecks. Also, “Synchronizing Flush” operations can be initiated – useful for ensuring the completion (vs. only parsing) of rendering instructions.
DefaultValue=0000 0000h

Programming Notes

All reserved bits are implemented.

DWord	Bit	Description
0	31:16	**Masks** Format: Mask[15:0] These bits serve as write enables for bits 15:0. If this register is written with any of these bits clear the corresponding bit in the field 15:0 will not be modified. Reading these bits always returns 0s.
	15:11	**Reserved** Project: All Format: MBZ
	10	**Reserved** Format: MBZ
	9	**TLB Invalidate** Format: U1 If set, this bit allows the command stream engine to invalidate the MFX TLBs. This bit is valid only with the Sync flush enable. *Note: GFX soft resets do not invalidate TLBs, it is up to GFX driver to explicitly invalidate TLBs post reset./*
	8:7	**Reserved** Format: MBZ
	6	**Memory Sync Enable** If set, this bit allows the video decode engine to write out the data from the local caches to memory.
	5	**Sync Flush Enable** Format: Enable (Cleared by HW)

VCS_INSTPM - VCS Instruction Parser Mode Register

		This field is used to request a Sync Flush operation. The device will automatically clear this bit before completing the operation. See Sync Flush *(Programming Environment)*. Setting the Sync Flush Enable will cause a config write to MMIO register space with the address 0x4f100. **Programming Notes** The command parser must be stopped prior to issuing this command by setting the **Stop Ring** bit in register **BCS_MI_MODE**. Only after observing **Ring Idle**set in **BCS_MI_MODE** can a Sync Flush be issued by setting this bit. Once this bit becomes clear again, indicating flush complete, the command parser is re-enabled by clearing **Stop Ring**.
	4:0	**Reserved** Access: R/W Format: MBZ

1.1.3.6 VCS_NOPID — NOP Identification Register

VCS_NOPID - VCS NOP Identification Register

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Address:	12094h-12097h

The VCS_NOPID register contains the Noop Identification value specified by the last MI_NOOP instruction that enabled this register to be updated.

DWord	Bit	Description
0	31:22	**Reserved** Format: MBZ
	21:0	**Identification Number** This field contains the 22-bit Noop Identification value specified by the last MI_NOOP instruction that enabled this field to be updated. **Programming Notes** • • Although this is a R/W register, it should only be written to by the MI_NOOP command. Write access is needed for power management support.

1.1.3.7 VBSYNC – Video/Blitter Semaphore Sync Register

VBSYNC - Video/Blitter Semaphore Sync Register

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	12040h

This register is written by BCS, read by VCS.

DWord	Bit	Description
0	31:0	**Semaphore Data** Semaphore data for synchronization between video codec engine and blitter engine.

1.1.3.8 VRSYNC – Video/Render Semaphore Sync Register

VRSYNC - Video/Render Semaphore Sync Register

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	12044h

This register is written by CS, read by VCS.

DWord	Bit	Description
0	31:0	**Semaphore Data** Semaphore data for synchronization between video codec engine and render engine.

1.1.3.9 GAC_MODE — Mode Register for GAC

GAC_MODE - Mode Register for GAC

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Address:	120A0h-120A3h

The GAC_MODE register contains information that controls configurations in the GAC.

DWord	Bit	Description
0	31:16	**Masks** Format: Mask[15:0] A 1 in a bit in this field allows the modification of the corresponding bit in Bits 15:0.
	15:0	**Reserved** Access: R/W

1.1.3.10 VCS_PREEMPTION_HINT_UDW

VCS_PREEMPTION_HINT_UDW - VCS_PREEMPTION_HINT_UDW

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Address:	124C8h

This register contains the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space of the Batch Buffer corresponding to MI_ARB_CHECK command called Preemption Hint Address.

Programming Notes
Programming Restriction: **This register should NEVER be programmed in functional mode, this should be used only in validation mode to achieve deterministic behavior of UHPTR being sampled by a given MI_ARB_CHK in command stream.**

DWord	Bit	Description
0	31:16	**Reserved** Format: MBZ
	15:0	**Preempted Hint Address** Format: GraphicsAddress[47:32] This field contains the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space of the batch buffer when Preemption Hint is set to Batch Buffer. This field is not valid when Preemption Hint is set to Ring Buffer.

1.1.4 VCS_RINGBUF—Ring Buffer Registers

RING_BUFFER_TAIL - Ring Buffer Tail

Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Access:	R/W

Address:	02030h
Name:	RCS Ring Buffer Tail
ShortName:	RCS_RING_BUFFER_TAIL
Address:	12030h
Name:	VCS Ring Buffer Tail
ShortName:	VCS_RING_BUFFER_TAIL
Address:	22030h
Name:	BCS Ring Buffer Tail
ShortName:	BCS_RING_BUFFER_TAIL

These registers are used to define and operate the “ring buffer” mechanism which can be used to pass instructions to the command interface. The buffer itself is located in a linear memory region. The ring buffer is defined by a 4 Dword register set that includes starting address, length, head offset, tail offset, and control information.

Refer to the Programming Interface chapter for a detailed description of the parameters specified in this ring buffer register set, restrictions on the placement of ring buffer memory, arbitration rules, and in how the ring buffer can be used to pass instructions.

Ring Buffer Tail Offsets must be properly programmed before ring is enabled. A Ring Buffer can be enabled when empty.

DWord	Bit	Description
0	31:21	**Reserved** Format: MBZ
	20:3	**Tail Offset** Format: GraphicsAddress[20:3] This field is written by software to specify where the valid instructions placed in the ring buffer end. The value written points to the QWord past the last valid QWord of instructions. In other words, it can be defined as the next QWord that software will write instructions into. Software must write subsequent instructions to QWords following the Tail Offset, possibly wrapping around to the top of the buffer (i.e., software can’t skip around within the buffer). Note that all DWords prior to the location indicated by the **Tail Offset** must contain valid instruction data – which may require instruction padding by software. See **Head Offset** for more information.
	2:0	**Reserved** Format: MBZ

RING_BUFFER_HEAD - Ring Buffer Head

Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Access:	R/W

Address:	02034h
Name:	RCS Ring Buffer Head
ShortName:	RCS_RING_BUFFER_HEAD
Address:	12034h
Name:	VCS Ring Buffer Head
ShortName:	VCS_RING_BUFFER_HEAD
Address:	22034h
Name:	BCS Ring Buffer Head
ShortName:	BCS_RING_BUFFER_HEAD

This register is used to define and operate the ring buffer mechanism which can be used to pass instructions to the command interface. The buffer itself is located in a physical memory region. The ring buffer is defined by a 4 Dword register set that includes starting address, length, head offset, tail offset, and control information. Refer to the Programming Interface chapter for a detailed description of the parameters specified in this ring buffer register set, restrictions on the placement of ring buffer memory, arbitration rules, and in how the ring buffer can be used to pass instructions.

Ring Buffer Head Offsets must be properly programmed before ring is enabled. A Ring Buffer can be enabled when empty.

DWord	Bit	Description
0	31:21	**Wrap Count** Format: U11 count of ring buffer wraps This field is incremented by 1 whenever the **Head Offset** wraps from the end of the buffer back to the start (i.e., whenever it wraps back to 0). Appending this field to the **Head Offset** field effectively creates a virtual 4GB Head “Pointer” which can be used as a tag associated with instructions placed in a ring buffer. The Wrap Count itself will wrap to 0 upon overflow.
	20:2	**Head Offset** Format: GraphicsAddress[20:2] DWord Offset This field indicates the offset of the *next* instruction DWord to be parsed. Software will initialize this field to select the first DWord to be parsed once the RB is enabled. (Writing the Head Offset while the RB is enabled is UNDEFINED). Subsequently, the device will increment this offset as it executes instructions – until it reaches the QWord specified by the **Tail Offset**. At this point the ring buffer is considered “empty”. **Programming Notes** A RB can be enabled empty or containing some number of valid instructions.
	1	**Reserved** Format: MBZ
	0	**Wait for Condition Indicator** Source: RenderCS

RING_BUFFER_HEAD - Ring Buffer Head

		This is a read only value used to indicate whether or not the command streamer is currently waiting for a conditional code to be cleared from 0x2028	
	0	**Reserved**	
		Source:	BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS
		Format:	MBZ

RING_BUFFER_START - Ring Buffer Start

Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Access:	R/W
Address:	02038h
Name:	RCS Ring Buffer Start
ShortName:	RCS_RING_BUFFER_START
Address:	12038h
Name:	VCS Ring Buffer Start
ShortName:	VCS_RING_BUFFER_START
Address:	22038h
Name:	BCS Ring Buffer Start
ShortName:	BCS_RING_BUFFER_START

These registers are used to define and operate the “ring buffer” mechanism which can be used to pass instructions to the command interface. The buffer itself is located in a physical memory region. The ring buffer is defined by a 4 Dword register set that includes starting address, length, head offset, tail offset, and control information. Refer to the Programming Interface chapter for a detailed description of the parameters specified in this ring buffer register set, restrictions on the placement of ring buffer memory, arbitration rules, and in how the ring buffer can be used to pass instructions.

DWord	Bit	Description	
0	31:12	**Starting Address**	
		Format:	GraphicsAddress[31:12]RingBuffer
		This field specifies Bits 31:12 of the 4KB-aligned starting Graphics Address of the ring buffer. Address bits 31 down to 29 must be zero. All ring buffer pages must map to Main Memory (uncached) pages. Ring Buffer addresses are always translated through the global GTT.	
	11:0	**Reserved**	
		Format:	MBZ

RING_BUFFER_CTL - Ring Buffer Control

Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Access:	R/W

Address:	0203Ch
Name:	RCS Ring Buffer Control
ShortName:	RCS_RING_BUFFER_CTL
Address:	1203Ch
Name:	VCS Ring Buffer Control
ShortName:	VCS_RING_BUFFER_CTL
Address:	2203Ch
Name:	BCS Ring Buffer Control
ShortName:	BCS_RING_BUFFER_CTL

These registers are used to define and operate the ring buffer mechanism which can be used to pass instructions to the command interface. The buffer itself is located in a physical memory region. The ring buffer is defined by a 4 Dword register set that includes starting address, length, head offset, tail offset, and control information. Refer to the Programming Interface chapter for a detailed description of the parameters specified in this ring buffer register set, restrictions on the placement of ring buffer memory, arbitration rules, and in how the ring buffer can be used to pass instructions.

Ring Buffer Head and Tail Offsets must be properly programmed before it is enabled. A Ring Buffer can be enabled when empty.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:21</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>20:12</td><td><b>Buffer Length</b><table><tr><td>Format:</td><td>U9-1 in 4 KB pages – 1</td></tr></table>This field is written by SW to specify the length of the ring buffer in 4 KB Pages.Range = [0 = 1 page = 4 KB, 1FFh = 512 pages = 2 MB]<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td></td><td>1 page = 4 KB</td></tr><tr><td>1FFh</td><td></td><td>512 pages = 2 MB</td></tr></table></td></tr>
<tr><td></td><td>11</td><td><b>RBWait</b>
Indicates that this ring has executed a WAIT_FOR_EVENT instruction and is currently waiting. Software can write a “1” to clear this bit, write of “0” has no effect. When the RB is waiting for an event and this bit is cleared, the wait will be terminated and the RB will be returned to arbitration.</td></tr>
<tr><td></td><td>10</td><td><b>Semaphore Wait</b><table><tr><th>Description</th><th>Project</th></tr><tr><td>Indicates that this ring has executed a MI_SEMAPHORE_MBOX instruction with register compare and is currently waiting.</td><td></td></tr></table></td></tr>
<tr><td></td><td>9</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>8</td><td><b>Reserved</b><table><tr><td>Source:</td><td>RenderCS, BlitterCS</td></tr></table></td></tr>
</table>

RING_BUFFER_CTL - Ring Buffer Control

Format:	MBZ

8

Disable Register Accesses

Source:	VideoCS, VideoCS2, VideoEnhancementCS

Value	Name	Description
0	R/W	Ring is allowed to access (read or write) MMIO space.
1	Read Only	Ring is not allowed to write MMIO space. Ring **is** allowed to read registers.

7:3

Reserved

Format:	MBZ

2:1

Automatic Report Head Pointer

Source:	BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS

Description	Project
This field is written by software to control the automatic “reporting” (write) of this ring buffer’s “Head Pointer” register (register DWord 1) to the corresponding location within the Hardware Status Page. Automatic reporting can either be disabled or enabled at 4KB, 64KB or 128KB boundaries within the ring buffer.	
The head pointer will be reported to the head pointer location in the Per-Process Hardware Status Page when it passes each 4KB page boundary. When the above-mentioned bit is set, reporting will behave just as on the prior devices (as documented above), and option 2 is not legal.	

Value	Name	Description
0	MI_AUTOREPORT_OFF	Automatic reporting disabled
1	MI_AUTOREPORT_64KB	Report every 16 pages (64KB)
2	MI_AUTOREPORT_4KB	Report every page (4KB)This mode must not be enabled in Ring Buffer mode of scheduling to minimize the auto reports.
3	MI_AUTOREPORT_128KB	Report every 32 pages (128KB)

2:1

Reserved

Source:	RenderCS
Format:	MBZ

0

Ring Buffer Enable

Format:	Enable

This field is used to enable or disable this ring buffer. It can be enabled or disabled regardless of whether there are valid instructions pending. If disabled and the ring head equals ring tail, all state currently loaded in hardware is considered invalid.

Programming Notes	Project
SW should follow the below programming notes while enabling render engine's ring buffer for the first time, this would be coming out of boot, standby, hibernate or reset. SW should set the Force Wakeup bit to prevent GT from entering C6. SW should dispatch workload (dummy) to initialize render engine with default state such that any context switches that occur subsequently (Power Save) will save and restore coherent	

RING_BUFFER_CTL - Ring Buffer Control			
		device state. Indirect pointers used in 3D states should point to valid graphics surface existing in memory. PP_DCLV followed by PP_DIR_BASE register should be programmed as part of initialization workload if PPGTT is enabled in GFX_MODE register. Once the render engine is programmed with valid state and the configuration, Force Wakeup bit should be reset to enable C6 entry.	

1.1.4.1 VCS_UHPTR — Pending Head Pointer Register

UHPTR - Pending Head Pointer Register	
Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Access:	R/W
Address:	02134h
Name:	RCS Pending Head Pointer Register
ShortName:	RCS_UHPTR
Address:	12134h
Name:	VCS Pending Head Pointer Register
ShortName:	VCS_UHPTR
Address:	22134h
Name:	BCS Pending Head Pointer Register
ShortName:	BCS_UHPTR

Programming Notes
Once SW uses UHPTR to preempt the existing workload, should explicitly program MI_SET_CONTEXT to save the preempted context status before submitting the new workload. In case SW doesn’t want to save the state of the preempted context, it should at the minimum program RS_PREEMPT_STATUS to 0x0 so that the register status doesn’t interfere with the new workloads.

DWord	Bit	Description
0	31:3	**Head Pointer Address** Format: GraphicsAddress[31:3] This register represents the GFX address offset where execution should continue in the ring buffer following execution of an MI_ARB_CHECK command.
	2:1	**Reserved** Format: MBZ
	0	**Head Pointer Valid** This bit is set by the software to request a pre-emption. It is reset by hardware when an MI_ARB_CHECK command is parsed by the command streamer. The hardware uses the head pointer programmed in this register at the time the reset is generated.

Value	Name	Description
0	InValid	No valid updated head pointer register, resume execution at the current location in the

UHPTR - Pending Head Pointer Register

			ring buffer
	1	Valid	Indicates that there is an updated head pointer programmed in this register

1.1.5 Watchdog Timer Registers

1.1.5.1 VCS_CNTR—Counter for the bit stream decode engine

VCS_CNTR - VCS Counter for the bit stream decode engine

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0xFFFFFFFF
Access:	R/W
Size (in bits):	32
Address:	12178h-1217Bh

DWord	Bit	Description
0	31:0	**Count Value** Default Value: ffffffffh Writing a Zero value to this register starts the counting. Writing a Value of FFFF FFFF to this counter stops the counter.

1.1.5.2 VCS_THRSH—VCS Threshold for the counter of bit stream decode engine

VCS_THRSH - VCS Threshold for the counter of bit stream decode engine

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00150000
Access:	R/W
Size (in bits):	32
Address:	1217Ch-1217Fh

DWord	Bit	Description
0	31:0	**Threshold Value** Default Value: 00150000h The value in this register reflects the number of clocks the bit stream decode engine is expected to run. If the value is exceeded the counter is reset and an interrupt may be enabled in the device.

1.1.6 Interrupt Control Registers

The Interrupt Control Registers described below all share the same bit definition. The bit definition is as follows:

Bit Definition for Interrupt Control Registers

Bit	Description
31:21	**Reserved. MBZ:** These bits may be assigned to interrupts on future products/steppings.
20	**Context Switch Interrupt:** Set when a context switch has just occurred.
19	**Page Fault:** This bit is set whenever there is a pending page or directory fault. This bit is set whenever there is a pending page or directory fault in Video command streamer.
18	**Timeout Counter Expired:** Set when the VCS timeout counter has reached the timeout thresh-hold value.
17	**Reserved**
16	**MI_FLUSH_DW Notify Interrupt:** 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.
15	**Video Command Parser Master Error:** 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 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

Bit	Description
	the interrupt to occur. **Page Table Error:** Indicates a page table error. **Instruction Parser Error**: The Video Instruction Parser encounters an error while parsing an instruction.
14	**Sync Status:** This bit is set when the Instruction Parser completes a flush with the sync enable bit active in the INSTPM register. The event will happen after all the MFX engines are flushed. The HW Status DWord write resulting from this event will cause the CPU's view of graphics memory to be coherent as well (flush and invalidate the MFX cache).It is the driver's responsibility to clear this bit before the next sync flush with HWSP write enabled
13	**Reserved: MBZ**
12	**Video Command Parser User Interrupt:** This status bit is set when an MI_USER_INTERRUPT instruction is executed on the Video 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.
11:0	**Reserved:** MBZ

The following table specifies the settings of interrupt bits stored upon a "Hardware Status Write" due to ISR changes:

Bit	Interrupt Bit	ISR bit Reporting via Hardware Status Write (when unmasked via HWSTAM)
8	**Context Switch Interrupt:** Set when a context switch has just occurred.	Not supported to be unmasked
7	**Page Fault:** This bit is set whenever there is a pending PPGTT (page or directory) fault.	Set when event occurs, cleared when event cleared
6	**Media Decode Pipeline Counter Exceeded Notify Interrupt:** The counter threshold for the execution of the media pipeline is exceeded. Driver needs to attempt hang recovery.	Not supported to be unmasked
5	**Reserved**	
4	MI_FLUSH_DW packet - Notify Enable	0
3	Master Error	Set when error occurs, cleared when error cleared
2	Sync Status	Set every SyncFlush Event
0	User Interrupt	0

1.1.6.1 VCS_HWSTAM - VCS Hardware Status Mask Register

VCS_HWSTAM - VCS Hardware Status Mask Register

Register Space:	MMIO: 0/2/0
Project:	All
Source:	VideoCS
Default Value:	0xFFFFFFFF
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	12098h

Access: RO for Reserved Control bits

The HWSTAM register has the same format as the Interrupt Control Registers. The bits in this register are “mask” bits that prevent the corresponding bits in the Interrupt Status Register from generating a “Hardware Status Write” (PCI write cycle). Any unmasked interrupt bit (HWSTAM bit set to 0) will allow the Interrupt Status Register to be written to the ISR location (within the memory page specified by the Hardware Status Page Address Register) when that Interrupt Status Register bit changes state.

Programming Notes

- To write the interrupt to the HWSP, the corresponding IMR bit must also be clear (enabled).
- At most 1 bit can be unmasked at any given time.

DWord	Bit	Description
0	31:0	**Hardware Status Mask Register** Default Value: FFFFFFFFh Format: Array of Masks Refer to the table in the Interrupt Control Register section for bit definitions.

1.1.6.2 VCS_IMR - VCS Interrupt Mask Register

VCS_IMR - VCS Interrupt Mask Register

Register Space:	MMIO: 0/2/0
Project:	All
Source:	VideoCS
Default Value:	0xFFFFFFFF
Access:	R/W
Size (in bits):	32
Address:	120A8h

The IMR register is used by software to control which Interrupt Status Register bits are masked or unmasked. Unmasked bits will be reported in the IIR, possibly triggering a CPU interrupt, and will persist in the IIR until cleared by software. Masked bits will not be reported in the IIR and therefore cannot generate CPU interrupts.

DWord	Bit	Description
0	31:0	**Interrupt Mask Bits** Format: Array of interrupt mask bits Refer to the Interrupt Control Register section for bit definitions. This field contains a bit mask which selects which interrupt bits (from the ISR) are reported in the IIR.

Value	Name	Description
FFFF FFFFh	**[Default]**	
0h	Not Masked	Will be reported in the IIR
1h	Masked	Will not be reported in the IIR

1.1.6.3 VCS Hardware - Detected Error Bit Definitions (for EIR, EMR, ESR)

This section defines the Hardware-Detected Error bit definitions and ordering that is common to the EIR, EMR and ESR registers. The EMR selects which error conditions (bits) in the ESR are reported in the EIR. Any bit set in the EIR will cause the Master Error bit in the ISR to be set. EIR bits will remain set until the appropriate bit(s) in the EIR is cleared by writing the appropriate EIR bits with ‘1’(except for the unrecoverable bits described below).

The following table describes the Hardware-Detected Error bits:

VCS Hardware-Detected Error Bit Definitions

Source:	VideoCS
Default Value:	0x00000000

DWord	Bit	Description
0	15:3	**Reserved** Format: MBZ
	2	**Reserved** Format: MBZ

VCS Hardware-Detected Error Bit Definitions

	1	**Reserved** Format: MBZ
	0	**Instruction Error** This bit is set when the Renderer Instruction Parser detects an error while parsing an instruction. Instruction errors include: Client ID value (Bits 31:29 of the Header) is not supported (only MI, 2D and 3D are supported). Defeatured MI Instruction Opcodes:

Value	Name	Description
1		Instruction Error detected

Programming Notes
This error indications cannot be cleared except by reset (i.e., it is a fatal error).

1.1.6.3.1 VCS_EIR — Error Identity Register

VCS_EIR - VCS Error Identity Register

Register Space:	MMIO: 0/2/0
Project:	All
Source:	VideoCS
Default Value:	0x00000000
Access:	R/WC
Size (in bits):	32
Address:	120B0h

The EIR register contains the persistent values of Hardware-Detected Error Condition bits. Any bit set in this register will cause the Master Error bit in the ISR to be set. The EIR register is also used by software to clear detected errors (by writing a 1 to the appropriate bit(s) except for the unrecoverable bits described).

DWord	Bit	Description
0	31:16	**Reserved** Format: MBZ
	15:0	**Error Identity Bits** Format: Array of Error condition bits ee the table titled Hardware-Detected Error Bits This register contains the persistent values of ESR error status bits that are unmasked via the EMR register. The logical OR of all (defined) bits in this register is reported in the Master Error bit of the Interrupt Status Register. In order to clear an error condition, software must first clear the error by writing a 1 to the appropriate bit(s) in this field. If required, software should then proceed to clear the Master Error bit of the IIR.

Value	Name	Description

VCS_EIR - VCS Error Identity Register

0h	**[Default]**	
1h	Error occurred	Error occurred

Programming Notes

Writing a 1 to a set bit will cause that error condition to be cleared. However, the Page Table Error bit (Bit 4) cannot be cleared except by reset (i.e., it is a fatal error).

1.1.6.3.2 VCS_EMR - VCS Error Mask Register

VCS_EMR - VCS Error Mask Register

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x0000FFFF
Access:	R/W
Size (in bits):	32
Address:	120B4h

The EMR register is used by software to control which Error Status Register bits are “masked” or “unmasked”. “Unmasked” bits will be reported in the EIR, thus setting the Master Error ISR bit and possibly triggering a CPU interrupt, and will persist in the EIR until cleared by software. “Masked” bits will not be reported in the EIR and therefore cannot generate Master Error conditions or CPU interrupts.

DWord	Bit	Description
0	31:16	**Reserved** Format: MBZ
	15:0	**Error Mask Bits** Format: Array of error condition mask bits See the table titled Hardware-Detected Error Bits. This register contains a bit mask that selects which error condition bits (from the ESR) are reported in the EIR.

Value	Name	Description
FFFF FFFFh	**[Default]**	
0h	Not Masked	Will be reported in the EIR
1h	Masked	Will not be reported in the EIR

1.1.6.3.3 VCS_ESR - VCS Error Status Register

VCS_ESR - VCS Error Status Register

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Address:	120B8h

The ESR register contains the current values of all Hardware-Detected Error condition bits (these are all by definition persistent). The EMR register selects which of these error conditions are reported in the persistent EIR (i.e., set bits must be cleared by software) and thereby causing a Master Error interrupt condition to be reported in the ISR.

<table>
<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></td></tr>
<tr><td></td><td>15:0</td><td><b>Error Status Bits</b><table><tr><td>Format:</td><td>Array of error condition bits See the table titled Hardware-Detected Error Bits.</td></tr></table>This register contains the non-persistent values of all hardware-detected error condition bits.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td><b>[Default]</b></td><td></td></tr><tr><td>1h</td><td>Error Condition Detected</td><td>Error Condition detected</td></tr></table></td></tr>
</table>

1.1.7 Logical Context Support

1.1.7.1 BB_ADDR—Batch Buffer Head Pointer Register

BB_ADDR - Batch Buffer Head Pointer Register

Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Address:	02140h
Name:	RCS Batch Buffer Head Pointer Register
ShortName:	RCS_BB_ADDR
Address:	12140h
Name:	VCS Batch Buffer Head Pointer Register
ShortName:	VCS_BB_ADDR
Address:	1A140h
Name:	VECS Batch Buffer Head Pointer Register
ShortName:	VECS_BB_ADDR
Address:	22140h
Name:	BCS Batch Buffer Head Pointer Register
ShortName:	BCS_BB_ADDR

This register contains the current DWord Graphics Memory Address of the last-initiated batch buffer.

Programming Notes

Programming Restriction: This register should NEVER be programmed by driver. This is for HW internal use only.

DWord	Bit	Description
0	31:3	**Batch Buffer Head Pointer** Source: BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS Format: GraphicsAddress[31:3] This field specifies the DWord-aligned Graphics Memory Address where the last initiated Batch Buffer is currently fetching commands. If no batch buffer is currently active, the Valid bit will be 0 and this field will be meaningless.
	31:2	**Batch Buffer Head Pointer** Source: RenderCS Format: GraphicsAddress[31:2] This field specifies the DWord-aligned Graphics Memory Address where the last initiated Batch Buffer is currently fetching commands. If no batch buffer is currently active, the Valid bit will be 0 and this field will be meaningless.
	2	**Reserved**

BB_ADDR - Batch Buffer Head Pointer Register

		Source:	BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS
		Format:	MBZ
	1	**Reserved**	
		Format:	MBZ
	0	**Valid**	
		Format:	U1

Value	Name	Description
0h	Invalid **[Default]**	Batch buffer Invalid
1h	Valid	Batch buffer Valid

1.1.7.2 BB_STATE - Batch Buffer State Register

BB_STATE - Batch Buffer State Register

Register Space:	MMIO: 0/2/0
Source:	BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS
Default Value:	0x00000000 [IVB:GT1]
Access:	RO
Size (in bits):	32

Address:	12110h
Name:	VCS Batch Buffer State Register
ShortName:	VCS_BB_STATE

Address:	1A110h
Name:	VECS Batch Buffer State Register
ShortName:	VECS_BB_STATE

Address:	22110h
Name:	BCS Batch Buffer State Register
ShortName:	BCS_BB_STATE

This register contains the attributes of the current batch buffer initiated from the Ring Buffer. These include the security indicator.

This register should not be written by software. These fields should only get written by a context restore. Software should always set these fields via the MI_BATCH_BUFFER_START command when initiating a batch buffer.

This register is saved and restored with context.

DWord	Bit	Description	
0	31:7	**Reserved**	
		Project:	All
		Format:	MBZ
	6	**2nd Level Buffer Security Indicator**	

BB_STATE - Batch Buffer State Register

<table>
<tr><td></td><td></td><td>
<table>
<tr><td></td><td></td></tr>
<tr><td>Source:</td><td>VideoCS, VideoCS2</td></tr>
</table>
If set, VCS is fetching 2nd level batch commands from a PPGTT address space. If clear, GGTT. If clear, this batch buffer is secure and will be accessed via the GGTT.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td><b>[Default]</b></td><td></td></tr>
<tr><td>0h</td><td>MIBUFFER_SECURE <b>[Default]</b></td><td>Located in GGTT memory</td></tr>
<tr><td>1h</td><td>MIBUFFER_NONSECURE</td><td>Located in PPGTT memory</td></tr>
</table>
</td></tr>
<tr><td></td><td>6</td><td><b>Reserved</b>
<table>
<tr><td></td><td></td></tr>
<tr><td>Source:</td><td>VideoCS, VideoCS2</td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
</table>
</td></tr>
<tr><td></td><td>6</td><td><b>Reserved</b>
<table>
<tr><td></td><td></td></tr>
<tr><td>Source:</td><td>BlitterCS, VideoEnhancementCS</td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
</table>
</td></tr>
<tr><td></td><td>5</td><td><b>1st Level Buffer Security Indicator</b>
<table>
<tr><td></td><td></td></tr>
<tr><td>Format:</td><td>MI_1stBufferSecurityType</td></tr>
</table>
If set, BCS is fetching 1st level batch commands from a PPGTT address space. If clear, GGTT.
It will be accessed via the PPGTT. If clear,
this batch buffer is secure and will be accessed via the GGTT.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>MIBUFFER_SECURE <b>[Default]</b></td><td>Located in GGTT memory</td></tr>
<tr><td>1h</td><td>MIBUFFER_NONSECURE</td><td>Located in PPGTT memory</td></tr>
</table>
</td></tr>
<tr><td></td><td>4</td><td><b>Reserved</b>
<table>
<tr><td>Project:</td><td>All</td></tr>
<tr><td>Source:</td><td>BlitterCS</td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
</table>
</td></tr>
<tr><td></td><td>3:0</td><td><b>Reserved</b>
<table>
<tr><td>Project:</td><td>All</td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
</table>
</td></tr>
</table>

1.1.8 Image Enhancement Registers

These registers contain the statistical data collected by Image Enhancement filters in Sampler (The denoise, deinterlace and film mode detection filter block) and Render Cache (The color enhancement filter block).

1.1.8.1 Denoise, Deinterlace and FMD Registers

Image Enhancement MMIO Registers

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000
Access:	RO
Size (in bits):	26x32
Trusted Type:	1
Address:	05000h

Address Offset:

05000h – 05028h: FMD Variances for Video Stream 0

0502Ch: GNE for Video Stream 0

05030h: Number of Valid GNE Blocks for Stream 0

05034h – 0505Ch: FMD Variances for Video Stream 1

05060h: GNE for Video Stream 1

05064h: Number of Valid GNE Blocks for Stream 1

The Denoise and Deinterlace features of Image Enhancement produce statistics across entire video frames for driver control of these features. Each of the supported video streams has a separate set of registers. The registers' contents are described in detail in Volume 5c Shared Functions, in the Deinterlacer and Denoise Filter section.

These registers are reset to zero when the read completes.

DWord	Bit	Description
0	31:0	**FMD Variance[0] for Video Stream 0** (For details, refer to the Simple Differences section in Volume 5c Shared Functions.)
1	31:0	**FMD Variance[1] for Video Stream 0**
2	31:0	**FMD Variance[2] for Video Stream 0**
3	31:0	**FMD Variance[3] for Video Stream 0**
4	31:0	**FMD Variance[4] for Video Stream 0**
5	31:0	**FMD Variance[5] for Video Stream 0**
6	31:0	**FMD Variance[6] for Video Stream 0**
7	31:0	**FMD Variance[7] for Video Stream 0**
8	31:0	**FMD Variance[8] for Video Stream 0**
9	31:0	**FMD Variance[9] for Video Stream 0**
10	31:0	**FMD Variance[10] for Video Stream 0**
11	31:0	**GNE Sum for Video Stream 0**

Image Enhancement MMIO Registers		
		(For details, refer to the Block Noise Estimate section in Volume 5c Shared Functions.)
12	31:0	**GNE Count for Video Stream 0**
13	31:0	**FMD Variance[0] for Video Stream 1** (For details, refer to the Simple Differences section in Volume 5c Shared Functions.)
14	31:0	**FMD Variance[1] for Video Stream 1**
15	31:0	**FMD Variance[2] for Video Stream 1**
16	31:0	**FMD Variance[3] for Video Stream 1**
17	31:0	**FMD Variance[4] for Video Stream 1**
18	31:0	**FMD Variance[5] for Video Stream 1**
19	31:0	**FMD Variance[6] for Video Stream 1**
20	31:0	**FMD Variance[7] for Video Stream 1**
21	31:0	**FMD Variance[8] for Video Stream 1**
22	31:0	**FMD Variance[9] for Video Stream 1**
23	31:0	**FMD Variance[10] for Video Stream 1**
24	31:0	**GNE Sum for Video Stream 1** (For details, refer to the Block Noise Estimate section in Volume 5c Shared Functions.)
25	31:0	**GNE Count for Video Stream 1**

1.1.8.2 Color Enhancement Registers

Color Enhancement Registers	
Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000
Access:	RO
Size (in bits):	48x32
Trusted Type:	1
Address:	05000h
Address Offset: 05080h: ACE Histogram Bin 0 (bits 15:0), Bin 1 (bits 31:16) 05084h to 517c: ACE Histogram Bins 2 through 127 (even bins in bits 15:0, odd bins in bits 31:16)	

Color Enhancement Registers

05070h: Skin Data Ymax (bits 25:16), Ymin (bits 9:0), other bits zero

05074h: Number of skin pixels (bits 20:0, other bits zero)

These registers are reset to zero when the read completes.

DWord	Bit	Description
0	31:16	**ACE Histogram Bin 1**
	15:0	**ACE Histogram Bin 0**
1..63	31:0	**ACE Histogram Bins 2 through 127** (even bins in bits 15:0, odd in 31:16)
64	31:26	**Reserved** Format: MBZ
	25:16	**Skin Data Ymax**
	15:10	**Reserved** Format: MBZ
	9:0	**Skin Data Ymin**

1.1.9 Registers in Media Engine

1.1.9.1 Introduction

This chapter describes the memory-mapped registers associated with the Memory Interface, including brief descriptions of their use. The functions performed by some of these registers are discussed in more detail in the Memory Interface Functions, Memory Interface Instructions, and Programming Environment chapters.

1.1.9.2 GAC PWR CTX STORAGE REGISTERS

1.1.9.2.1 GFX_PEND_TLB – Max Outstanding Media pending TLB requests

GFX_PEND_TLB - TLBPEND Control Register	
Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	14040h
Max Outstanding Media pending TLB requests	

DWord	Bit	Description
0	30	**Reserved** Format: MBZ
	29:24	**VMX BS Limit Count** Format: U6 This is the MAX number of Allowed internal pending read requests which require a TLB read.
	23	**VMC Limit Enable bit** Format: U1 This bit is used to enable the pending TLB requests limitation function for the VMC. When set, the number of internal pending read requests which require a TLB read will not exceed the programmed counter value.
	22	**Reserved** Format: MBZ
	21:16	**VMC TLB Limit Count** Format: U6 This is the MAX number of Allowed internal pending read requests which require a TLB read.
	15	**VMXRS Limit Enable bit** Format: U1 This bit is used to enable the pending TLB requests limitation function for the VMX Row store. When set, the number of internal pending read requests which require a TLB read will not exceed the programmed counter value.
	14	**Reserved** Format: MBZ
	13:8	**VMX RS Random Accsess TLB Limit Count** Format: U6

GFX_PEND_TLB - TLBPEND Control Register

		This is the MAX number of Allowed internal pending read requests which require a TLB read.
	7	**VCS Limit Enable bit** Format: U1 This bit is used to enable the pending TLB requests limitation function for the Command Streamer. When set, the number of internal pending read requests which require a TLB read will not exceed the programmed counter value.
	6	**Reserved** Format: MBZ
	5:0	**VCS TLB Limit Count** Format: U6 This is the MAX number of Allowed internal pending read requests which require a TLB read.

1.1.9.2.2 GAC_ARB_CTL_REG - GAC_GAB Arbitration Counters Register 1

GAC_ARB_CTL_REG - GAC_GAB Arbitration Counters Register 1

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00400002
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	14050h

GAC_GAB R/RO/W Arbitration Control Register

DWord	Bit	Description
0	31	**GAC write request Limit Enable** Format: U1 As long As there is no conflict between GAC and GAB ,GAC will alow whoever shows up (if media present and no GAB, let meda and vice versa). If both are present, start counting and switch when programmable no of request is expired. Allow only One GAB request and reset the counter. Counter only counts while we service a particular client and another client is present, else counter will reset.
	30	**VLF Final write Limit Enable** Format: MBZ As long as there is no conflict Between VCS MFD and VLF Final Write, GAC will allow whoever shows up (if VLF present and no VCSMFD, Let VLF and vice versa). If both are present, Start counting and when programmable no of request is expired. Allow only One VCSMFD request And counter will reset. Counter only counts while we service a particular client and another client is present, else counter will reset.

GAC_ARB_CTL_REG - GAC_GAB Arbitration Counters Register 1

<table>
<tr><td></td><td>29:24</td><td><b>Write Req Limit Count</b><table><tr><td>Format:</td><td>U6</td></tr></table>The value programmed determines the number of GAC/VLF Writes will allow for Each time.</td></tr>
<tr><td></td><td>23</td><td><b>GAC/GAB Cascaded Read Only Limit Enable</b><table><tr><td>Format:</td><td>U1</td></tr></table>As long as there is no conflict between GAC and GAB Read Requests, GAC will allow whoever shows up (if GAC present and no GAB, Let GAC and vice versa). If both are present, Start counting and switch when programmable no of request from either side is expired (reset the counter when switch). Counter only counts while we service a particular client and other client is present, else counter will reset.</td></tr>
<tr><td></td><td>22</td><td><b>Fixed Priority Setting</b><table><tr><td>Format:</td><td>MBZ</td></tr></table>Once programmable counter is disabled, GAC uses the fixed arbitration setting given in this register setting.<table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>GAC</td></tr><tr><td>1</td><td>GAB <b>[Default]</b></td></tr></table></td></tr>
<tr><td></td><td>21</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>20:16</td><td><b>GAC/GAB Read Only Limit Counter Value</b><table><tr><td>Format:</td><td>U6</td></tr></table>This is the Maximum number of Read requests Allowed from Each Cascaded Agent. Default 0</td></tr>
<tr><td></td><td>15</td><td><b>GAC/GAB Cascaded Read Limit Enable</b><table><tr><td>Format:</td><td>U1</td></tr></table><table><tr><td>As long as there is no conflict Between GAC and GAB Read Only Requests, GAC will allow whoever shows up (if GAC present and no GAB, Let GAC and vice versa). If both are present, Start counting and switch when programmable no of request from either side is expired (reset the counter when switch). Counter only counts while we service a particular client and other client is present, else counter will reset.</td></tr><tr><td>Default 0</td></tr></table></td></tr>
<tr><td></td><td>14</td><td><b>Default priority 0-GAC, 1-GAB</b><table><tr><td>Format:</td><td>MBZ</td></tr></table>Default 0</td></tr>
<tr><td></td><td>13</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>12:8</td><td><b>GAC/GAB Read Limit Counter Value</b><table><tr><td>Format:</td><td>U6</td></tr></table>This is the Maximum number of Read requests allowed from Each Cascaded Agent.</td></tr>
<tr><td></td><td>7:6</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>5:0</td><td><b>No of Global GTT Entries Valid in PPGTT mode in TLB064</b></td></tr>
</table>

GAC_ARB_CTL_REG - GAC_GAB Arbitration Counters Register 1

		Default Value: 000010b Format: U6 Minimum value the PPGGTT LRA can have (effectively partitioning the TLB between PPGTT and GGTT). Currently, only 2 entries are allocated to GGTT in ASmodel. TLB64 is shared by GGTT and PPGTT entries, are 2 LRAs, the GGTT one running from 0 up to PPGTT_MIN -1 (which is 2, but could be changed if needed), and the PPGTT one running from PPGTT_MIN up to 63.

1.1.9.3 GFX TLB In Use Virtual Address Registers

1.1.9.3.1 TLB064_VA — Virtual Page Address Registers

TLB064_VA - TLB064_VA Virtual Page Address Registers

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	14800h-148FCh

This register is directly mapped to the current Virtual Addresses in the TLB064 (VCS and VMC TLB).

DWord	Bit	Description
0	31:12	**Address** Format: GraphicsAddress[31:12] Page virtual address.
	11:0	**Reserved** Format: MBZ

1.1.9.3.2 TLB132_VA — Virtual Page Address Registers

TLB132_VA - TLB132_VA Virtual Page Address Registers

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Address:	14900h-149FCh

These registers are directly mapped to the current Virtual Addresses in the TLB132 (All The Media Clients TLB). Default Value = UUUUUUUUh Trusted Type = 1

DWord	Bit	Description
0	31:12	**Address** Format: GraphicsAddress[31:12] Page virtual address.
	11:0	**Reserved** Format: MBZ

1.1.9.3.3 TLB232_VA — Virtual Page Address Registers

TLB232_VA - TLB232_VA Virtual Page Address Registers

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	14A00h-14AFCh

This register is directly mapped to the current Virtual Addresses in the TLB232 (VDS and VLF FW TLB).

DWord	Bit	Description
0	31:12	**Address** Format: GraphicsAddress[31:12] Page virtual address.
	11:0	**Reserved** Format: MBZ

1.1.9.3.4 TLB304_VA — Virtual Page Address Registers

TLB304_VA - TLB304_VA Virtual Page Address Registers

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	14B00h-14BFCh

This register is directly mapped to the current Virtual Addresses in the TLB304 (VCR TLB).

DWord	Bit	Description
0	31:12	**Address** Format: GraphicsAddress[31:12] Page virtual address.
	11:0	**Reserved** Format: MBZ

1.1.9.3.5 TLB064_VLD — Valid Bit Vector 0 for TLB

MTTLB064_VLD0 - Valid Bit Vector 0 for TLB064

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	14780h-14783h

This register contains the valid bits for entries 0-31 of MTTLB (Texture and constant cache TLB).

DWord	Bit	Description
0	31:0	**Valid bits per entry**

1.1.9.3.6 TLB064_VLD — Valid Bit Vector 1 for TLB

MTTLB064_VLD1 - Valid Bit Vector 1 for TLB064

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	14784h-14787h

This register contains the valid bits for entries 0-31 of MTTLB (Texture and constant cache TLB).

DWord	Bit	Description
0	31:0	**Valid bits per entry**

1.1.9.3.7 TLB132_VLD — Valid Bit Vector 0 for TLB

MTTLB132_VLD0 - Valid Bit Vector 0 for TLB132

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	14788h-1478Bh

This register contains the valid bits for entries 0-31 of MTTLB (Texture and constant cache TLB).

DWord	Bit	Description
0	31:0	**Valid bits per entry**

1.1.9.3.8 TLB132_VLD — Valid Bit Vector 1 for MTTLB

MTTLB132_VLD1 - Valid Bit Vector 1 for TLB132		
Register Space:		MMIO: 0/2/0
Source:		VideoCS
Default Value:		0x00000000
Access:		RO
Size (in bits):		32
Trusted Type:		1
Address:	1478Ch-1478Fh	
This register contains the valid bits for entries 0-31 of MTTLB (Texture and constant cache TLB).		
DWord	**Bit**	**Description**
0	31:0	**Valid bits per entry**

1.1.9.3.9 TLB232_VLD — Valid Bit Vector 0 for TLB

MTTLB232_VLD0 - Valid Bit Vector 0 for TLB232		
Register Space:		MMIO: 0/2/0
Source:		VideoCS
Default Value:		0x00000000
Access:		RO
Size (in bits):		32
Trusted Type:		1
Address:	14790h-14793h	
This register contains the valid bits for entries 0-31 of MTTLB (Texture and constant cache TLB).		
DWord	**Bit**	**Description**
0	31:0	**Valid bits per entry**

1.1.9.3.10 TLB232_VLD — Valid Bit Vector 1 for MTTLB

MTTLB232_VLD1 - Valid Bit Vector 1 for TLB232

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	14794h-14797h

This register contains the valid bits for entries 0-31 of MTTLB (Texture and constant cache TLB).

DWord	Bit	Description
0	31:0	**Valid bits per entry**

1.1.9.3.11 TLB304_VLD — Valid Bit Vector 0 for TLB304

MTTLB304_VLD0 - Valid Bit Vector 0 for TLB304

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	14798h-1479Bh

This register contains the valid bits for entries 0-31 of MTTLB (Texture and constant cache TLB).

DWord	Bit	Description
0	31:0	**Valid bits per entry**

1.1.9.3.12 TLB304_VLD — Valid Bit Vector 1 for TLB304

MTTLB304_VLD1 - Valid Bit Vector 1 for TLB304

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	1479Ch-1479Fh

This register contains the valid bits for entries 0-31 of MTTLB (Texture and constant cache TLB). Default Value = 00000000h Trusted Type = 1

DWord	Bit	Description
0	31:0	**Valid bits per entry**

1.1.9.4 GFX Pending TLB Cycles Information Registers

The following registers contain information about cycles that did not complete their TLB translation.

Information is organized as 64 entries, where each entry has a valid and ready bit, collapsed into separate registers.

1.1.9.4.1 VCS_TLBPEND_VLD0 - VCS Valid Bit Vector 0 for TLBPEND Registers

VCS_TLBPEND_VLD0 - VCS Valid Bit Vector 0 for TLBPEND Registers

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	14700h-14703h

This register contains the valid bits for entries 0-31 of TLBPEND structure (Cycles pending TLB translation).

DWord	Bit	Description
0	31:0	**Valid bits per entry**

1.1.9.4.2 VCS_TLBPEND_VLD1 - VCS Valid Bit Vector 1 for TLBPEND Registers

VCS_TLBPEND_VLD1 - VCS Valid Bit Vector 1 for TLBPEND Registers

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	14704h-14707h

This register contains the valid bits for entries 32-63 of TLBPEND structure (Cycles pending TLB translation).

DWord	Bit	Description
0	31:0	**Valid bits per entry**

1.1.9.4.3 VCS_TLBPEND_RDY0 - VCS Ready Bit Vector 0 for TLBPEND Registers

VCS_TLBPEND_RDY0 - VCS Ready Bit Vector 0 for TLBPEND Registers

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	14708h-1470Bh

This register contains the ready bits for entries 0-31 of TLBPEND structure (Cycles pending TLB translation).

DWord	Bit	Description
0	31:0	**Ready bits per entry**

1.1.9.4.4 VCS_TLBPEND_RDY1 - VCS Ready Bit Vector 1 for TLBPEND Registers

VCS_TLBPEND_RDY1 - VCS Ready Bit Vector 1 for TLBPEND Registers

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	1470Ch-1470Fh

This register contains the ready bits for entries 32-63 of TLBPEND structure (Cycles pending TLB translation).

DWord	Bit	Description
0	31:0	**Ready bits per entry**

1.1.9.4.5 VCS_TLBPEND_SEC0 - VCS Section 0 of TLBPEND Entry

VCS_TLBPEND_SEC0 - VCS Section 0 of TLBPEND Entry

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	14400h-144FCh

This register is directly mapped to the TLBPEND Array in the Graphic Arbiter.

DWord	Bit	Description
0	31	**vtstatus** This bit will be used in conjunction with the ready bit to determine the stage of the translation. See table below.
	30:28	**GTT bits** Bits 3:1 of the GTT entry used to translate the Virtual Address. 000 if translation is pending.
	27:0	**Current address** The value of this field depends on the stage of the TLB translation for this entry: **VA** – bits 27:20 = 00, bits 19:0 = Bits 31:12 of the Virtual Address of the cycle.

VTDMODE	Valid	Ready	Vtstatus	Meaning
DC	0	DC	DC	Entry is invalid
0	1	0	0	Entry was a TLB miss. Waiting for TLB translation.
0	1	0	1	Entry was a Hit not present. Waiting for TLB translation from a previous miss.
0	1	1	0	Not possible
0	1	1	1	TLB translation complete. Entry ready
1	1	0	0	Entry was a TLB miss. Waiting for TLB translation.
1	1	0	1	Entry was a Hit not present. Waiting for TLB translation from a previous miss.
1	1	1	0	GPA translation complete. Entry ready for VTD translation.
1	1	1	1	TLB translation complete. Entry ready

1.1.9.4.6 TLBPEND_SEC1 — Section 1 of TLBPEND Entry

VCS_TLBPEND_SEC1 - VCS Section 1 of TLBPEND Entry

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	14500h-145FCh

This register is directly mapped to the current Virtual Addresses in the MTTLB (Texture and constant cache TLB).

DWord	Bit	Description	
0	31:28	**Current address** Bits 9:6 of the Virtual Address of the cycle.	
	27:24	**Cacheability Control Bits**	
		2	**Graphics Data Type (GFDT)**
		This field contains the GFDT bit for this surface when writes occur. GFDT can also be set by the GTT. The effective GFDT is the logical OR of this field with the GFDT from the GTT entry. This field is ignored for reads.	
		1:0	**Cacheability Control**
		This field controls cacheability in the mid-level cache (MLC) and last-level cache (LLC).	
		00: use cacheability control bits from GTT entry	
		01: data is not cached in LLC or MLC	
		10: data is cached in LLC but not MLC	
		11: data is cached in both LLC and MLC	
	23	**ZLR bit** Flag to indicate this is a zero length read (A read used to calculate a Physical Address for a write).	
	22:4	**TAG** Cycle identification TAG.	
	3:0	**SRC ID**	

VCS_TLBPEND_SEC1 - VCS Section 1 of TLBPEND Entry

Encoding of unit generating this cycle

Constant	Value
SRCID	
VCS_RD_SRCID	“00000”
VMC_RD_SRCID	“00001”
VMX_RARD_SRCID	“00010”
VMX_BSRD_SRCID	“00011”
VMX_RSRD_SRCID	“00100”
VIP_RD_SRCID	“00101”
VLF_RD_SRCID	“00110”
VDS_ZLRD_SRCID	“00111”
VCS_WR_SRCID	“01000”
VMX_BSWR_SRCID	“01001”
VDS_WR_SRCID	“01010”
VOP_WR_SRCID	“01011”
VLF_RSWR_SRCID	“01100”
VLF_FDWR_SRCID	“01101”
VMX_RSWR_SRCID	“01110”
BSP_WR_SRCID	“01111”
VCR_RD_SRCID	“10001”
VCR_WR_SRCID	“10010”
VCS_RD_PROBE	“10011”

1.1.9.4.7 VCS_TLBPEND_SEC2 - VCS Section 2 of TLBPEND Entry

VCS_TLBPEND_SEC2 - VCS Section 2 of TLBPEND Entry

Register Space:	MMIO: 0/2/0
Source:	VideoCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	14600h-146FCh

This register is directly mapped to the current Virtual Addresses in the MTTLB (Texture and constant cache TLB).

DWord	Bit	Description
0	31:11	**Reserved**
	10:8	**Current address** Bits 11:9 of the Virtual Address of the cycle.
	7:0	**PAT entry** Location of Physical Address in Physical Address Table.

1.1.9.5 VCS_TIMESTAMP - VCS Reported Timestamp Count

VCS_TIMESTAMP - VCS Reported Timestamp Count	
Register Space:	MMIO: 0/2/0
Project:	All
Source:	VideoCS
Default Value:	0x00000000, 0x00000000
Access:	RO. This register is not set by the context restore.
Size (in bits):	64
Address:	12358h

This register provides an elapsed real-time value that can be used as a timestamp. This register is not reset by a graphics reset. It will maintain its value unless a full chipset reset is performed.

Note: This timestamp register reflects the value of the PCU TSC. The PCU TSC counts 10ns increments; this timestamp reflects bits 38:3 of the TSC (i.e. 80ns granularity, rolling over every 1.5 hours).

DWord	Bit	Description
0	63:36	**Reserved** Format: MBZ
	35:0	**Timestamp Value** Format: U32 This register toggles every 80 ns. The upper 28 bits are zero.

1.2 Memory Interface Commands for Video Codec Engine

1.2.1 Introduction

This chapter describes the formats of the “Memory Interface” commands, including brief descriptions of their use. The functions performed by these commands are discussed fully in the *Memory Interface Functions* Device Programming Environment chapter.

This chapter describes MI Commands for the Video Codec Engine.

The commands detailed in this chapter are used across the later products within the architecture. However, slight changes may be present in some commands (i.e., for features added or removed), or some commands may be removed entirely. Refer to the *Preface* chapter for details.

1.2.2 MI_ARB_CHECK

MI_ARB_CHECK

Project:	All
Source:	VideoCS
Length Bias:	1

The MI_ARB_CHECK is used to check for a change in arbitration. If executed as part of a Ring Buffer the command checks the UHPTR valid bit and if set the head of the ring will jump to the value of the head pointer programmed in the UHPTR.

Programming Notes

This instruction cannot be placed in a batch buffer.

DWord	Bit	Description
0	31:29	**MI Instruction Type** Default Value: 0h MI_INSTRUCTION Format: OpCode
	28:23	**MI Instruction Opcode** Default Value: 05h MI_ARB_CHECK Format: OpCode
	22:0	**Reserved** Format: MBZ

1.2.3 MI_ARB_ON_OFF

MI_ARB_ON_OFF

Source:	VideoCS
Length Bias:	1

The MI_ARB_ON_OFF instruction is used to disable/enable context switching. Note that context switching will remain disabled until re-enabled through use of this command. This command will also prevent a switch in the case of running out of commands. This will effectively hang the device if allowed to occur while arbitration is off (context switching is disabled.) This command should always be used as an off-on pair with the sequence of instructions to be protected from context switch between MI_ARB_OFF and MI_ARB_ON.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND
	28:23	**MI Command Opcode** Default Value: 08h MI_ARB_ON_OFF
	22:1	**Reserved** Format: MBZ
	0	**Arbitration Enable** Format: Enable This field enables or disables context switches due to pre-emption. Value – Name 0h – Disabled 1h – Enabled

1.2.4 MI_BATCH_BUFFER_END

The MI_BATCH_BUFFER_END command format follows:

MI_BATCH_BUFFER_END	
Project:	All
Source:	VideoCS
Length Bias:	1

The MI_BATCH_BUFFER_END command is used to terminate the execution of commands stored in a batch buffer initiated using a MI_BATCH_BUFFER_START command.

DWord	Bit	Description	
0	31:29	**Command Type**	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	**MI Command Opcode**	
		Default Value:	0Ah MI_BATCH+_BUFFER_END
		Format:	OpCode
	22:0	**Reserved**	
		Format:	MBZ

1.2.5 MI_CONDITIONAL_BATCH_BUFFER_END

MI_CONDITIONAL_BATCH_BUFFER_END	
Source:	VideoCS
Length Bias:	2

The MI_BATCH_BUFFER_END command is used to conditionally terminate the execution of commands stored in a batch buffer initiated using a MI_BATCH_BUFFER_START command.

Programming Notes
This command is only valid with a 1st level batch buffer (bit 22 in MI_BATCH_BUFFER_START is set to 0).

DWord	Bit	Description	
0	31:29	**Command Type**	
		Default Value:	0h MI_COMMAND
	28:23	**MI Command Opcode**	
		Default Value:	36h MI_CONDITIONAL_BATCH_BUFFER_END
	22	**Use Global GTT**	
		Default Value:	0h DefaultVaueDesc
		Format:	U1
		Format:	U1 FormatDesc
		If set, this command will use the global GTT to translate the Compare Address and this command must be executing from a privileged (secure) batch buffer. If clear, the PPGTT will be used to translate the Compare Address.	

MI_CONDITIONAL_BATCH_BUFFER_END

	21	**Compare Semaphore** Default Value: 0h DefaultVaueDesc Format: U1 If set, the value from the Compare Data Dword is compared to the value from the Compare Address in memory. If the value at Compare Address is greater than the Compare Data Dword, execution of current command buffer should continue.If clear, no comparison takes place.
	19:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:0	**Compare Data Dword** Data dword to compare memory. The Data dword is supplied by software to control execution of the command buffer. If the compare is enabled and the data at Semaphore Address is greater than this dword, the execution of the command buffer should continue.
2	31:3	**Compare Address** Qword address to fetch compare Mask (DW0) and Data Dword(DW1) from memory. HW will do AND operation on Mask(DW0) with Data Dword(DW1) and then compare the result against Semaphore Data Dword
	2:0	**Reserved** Format: MBZ

1.2.6 MI_BATCH_BUFFER_START

The MI_BATCH_BUFFER_START command format follows:

MI_BATCH_BUFFER_START

Source:	VideoCS
Length Bias:	2

The MI_BATCH_BUFFER_START command is used to initiate the execution of commands stored in a batch buffer. For restrictions on the location of batch buffers, see Batch Buffers in the Device Programming Interface chapter of MI Functions.The batch buffer can be specified as secure or non-secure, determining the operations considered valid when initiated from within the buffer and any attached (chained) batch buffers. See Batch Buffer Protection in the Device Programming Interface chapter of MI Functions.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 31h MI_BATCH_BUFFER_START Format: OpCode

<table>
<tr><th colspan="3">MI_BATCH_BUFFER_START</th></tr>
<tr><td></td><td>22</td><td><b>2nd Level Batch Buffer</b>
<table><tr><td></td><td></td></tr></table>
The command streamer contains 3 storage elements; 1 for the ring head address, 1 for the batch head address, and 1 for the 2nd level batch head address. When performing batch buffer chaining, hardware simply updates the head pointer of the 1st level batch address storage. There is no stack in hardware. When this bit is set, hardware uses the 2nd level batch head address storage element. Upon MI_BATCH_BUFFER_END, it will automatically return to the 1st (traditional) level batch buffer address. this allows hardware to mimic a simple 3 level stack.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>1st level batch</td><td>Place the batch buffer address in the 1st (traditional) level batch address storage element</td></tr>
<tr><td>1h</td><td>2nd level batch</td><td>Place the batch buffer address in the 2nd level batch address storage element</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td><ul><li>A non-secure 2nd level batch buffer cannot be called from a non-secure 1st(traditional) level batch buffer.</li><li>2nd level batch buffer chaining is not supported.</li></ul></td></tr>
</table></td></tr>
<tr><td></td><td>21:10</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>8</td><td><b>Address Space Indicator</b>
<table>
<tr><td></td><td></td></tr>
<tr><td>Format:</td><td>U32</td></tr>
<tr><td>Format:</td><td>MI_BufferSecurityType</td></tr>
</table>
<table>
<tr><td>Certain operations (e.g., MI_STORE_DATA_IMM commands to privileged memory) are prohibited within non-secure buffers. See Batch Buffer Protection in the Device Programming Interface chapter of MI Functions. The command streamer will not allow a batch buffer in PPGTT to call a batch buffer in GGTT space by retaining the PPGTT value. It is illegal for the driver to program the value of this field to a different value than the current batch buffer executing this command.</td></tr>
<tr><td>This field must be 0 unless the Per-Process GTT Enable is 1.</td></tr>
</table>
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0</td><td>MIBUFFER_SECURE (GGTT space)</td></tr>
<tr><td>1</td><td>MIBUFFER_NONSECURE (PPGTT space)</td></tr>
</table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b>
<table>
<tr><td>Default Value:</td><td>0h Excludes DWord (0,1)</td></tr>
<tr><td>Format:</td><td>=n Total Length - 2</td></tr>
</table></td></tr>
<tr><td>1</td><td>31:2</td><td><b>Buffer Start Address</b>
<table><tr><td>Format:</td><td>GraphicsAddress[31:2]</td></tr></table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td><ul><li>A batch buffer initiated with this command must end either with a MI_BATCH_BUFFER_END</li></ul></td></tr>
</table></td></tr>
</table>

MI_BATCH_BUFFER_START

		command or by chaining to another batch buffer with an MI_BATCH_BUFFER_START command. • The selection of PPGTT vs. GGTT for the batch buffer is determined by the Buffer Security Indicator (bit8).
	1:0	**Reserved** Format: MBZ

1.2.7 MI_FLUSH_DW

MI_FLUSH_DW

Project:	All
Source:	VideoCS
Length Bias:	2

The MI_FLUSH_DW command is used to perform an internal “flush” operation. The parser pauses on an internal flush until all drawing engines have completed any pending operations. In addition, this command can also be used to:Flush any dirty data to memory. Invalidate the TLB cache inside the hardware Usage note: After this command is completed with a Store DWord enabled, CPU access to graphics memory will be coherent (assuming the Render Cache flush is not inhibited).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND
	28:23	**MI Command Opcode** Default Value: 26h MI_FLUSH_DW
	21	**Store Data Index** Format: U1 This field is valid only if the post-sync operation is not 0. If this bit is set, the store data address is actually an index into the hardware status page. If this bit is set, this command will index into the per-process hardware status page if executed from within a non-secure batch buffer and if the Per-Process Virtual Address Space is set. Else the Global HW status page is used.
	20:19	**Reserved** Format: MBZ
	18	**TLB Invalidate** Format: U1 Description / Project: If ENABLED, all TLBs will be invalidated once the flush operation is complete. This bit is only valid when the Post-Sync Operation field is a value of 1h or 3h. If GFX_MODE(0x229c) bit 13, this command will cause a config write to MMIO register space with the address 0x4f100.
	17	**Synchronize GFDT surface**

MI_FLUSH_DW

Format:	U1

If enabled, at the end of the current flush the last level cache is cleared of all the cachelines which have been marked with the special GFDT flags. Store DW must be enabled

16 **Reserved**

Format:	MBZ

15:14 **Post-Sync Operation**

BitFieldDesc

Value	Name	Description	Project
0h		No write occurs as a result of this instruction. This can be used to implement a “trap” operation, etc.	
1h		Write the QWord containing Immediate Data Low, High DWs to the Destination Address	
2h		Reserved	
3h		Write the TIMESTAMP register to the Destination Address with a granularity of 80ns. The upper 28 bits of the TIMESTAMP register are tied to ‘0’.	

Programming Notes

13:9 **Reserved**

Format:	MBZ

8 **Notify Enable**

Project:	
Format:	U1

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

7 **Video Pipeline Cache invalidate**

Format:	U1

Enable the invalidation of the video cache at the end of this flush

5:0 **DWord Length**

Format:	=n Total Length - 2

Value	Name	Project
2h	Excludes DWord (0,1) = 1 for DWord, 2 for QWord **[Default]**	

1 31:3 **Address**

Format:	GraphicsAddress[31:3]U28

This field specifies Bits 31:3 of the Address where the DWord or QWord will be stored. Note that the address can only be QWord aligned, irrespective of data size.

<table>
<tr><th colspan="3">MI_FLUSH_DW</th></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td>2</td><td><b>Destination Address Type</b>
Defines address space of Destination Address
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>PPGTT</td><td>Use PPGTT address space for DW write</td></tr>
<tr><td>1h</td><td>GGTT</td><td>Use GGTT address space for DW write</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>Ignored if “No write” is the selected in Operation.</td></tr>
</table></td></tr>
<tr><td></td><td>1:0</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td>2..3</td><td>31:0</td><td><b>Immediate Data</b>
<table>
<tr><td></td><td></td></tr>
<tr><td>Format:</td><td>U64</td></tr>
</table>
<table>
<tr><td>This field specifies the DWord value to be written to the targeted location. DW2 is the lower DW if QW is desired. Only valid when 15:14 in header is set to 1h</td></tr>
<tr><td>To avoid hitting a known hardware bug, drivers cannot send a QW write when bit 5 of the address is ‘1’</td></tr>
</table>
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>[0,FFFFFFFFh]</td><td></td></tr>
</table></td></tr>
</table>

1.2.8 MI_LOAD_REGISTER_IMM

The MI_LOAD_REGISTER_IMM command format is:

<table>
<tr><th colspan="3">MI_LOAD_REGISTER_IMM</th></tr>
<tr><td colspan="3"><table>
<tr><td>Project:</td><td>All</td></tr>
<tr><td>Source:</td><td>VideoCS</td></tr>
<tr><td>Length Bias:</td><td>2</td></tr>
</table></td></tr>
<tr><td colspan="3">The MI_LOAD_REGISTER_IMM command requests a write of up to a DWord constant supplied in the command to the specified Register Offset (i.e., offset into Memory-Mapped Register Range). The register is loaded before the next command is executed.</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b>
<table>
<tr><td>Default Value:</td><td>0h MI_COMMAND</td></tr>
<tr><td>Format:</td><td>OpCode</td></tr>
</table></td></tr>
<tr><td></td><td>28:23</td><td><b>MI Command Opcode</b>
<table>
<tr><td>Default Value:</td><td>22h MI_LOAD_REGISTER_IMM</td></tr>
<tr><td>Format:</td><td>OpCode</td></tr>
</table></td></tr>
<tr><td></td><td>22:12</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
</table>

MI_LOAD_REGISTER_IMM

	11:8	**Byte Write Disables** Format: Enable[4] (bit 8 corresponds to Data DWord [7:0]). Range: Must specify a valid register write operation If [11:8] is '1111b', then the register write will not occur. If [11:8] is '0000b', then the register DW will be updated. Any other value, the behavior will be specifically specified by the register or the behavior is undefined.
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:23	**Reserved** Format: MBZ
	22:2	**Register Offset** Format: MmioAddress[22:2] This field specifies bits [22:2] of the offset into the Memory Mapped Register Range (i.e., this field specifies a DWord offset).Mapped
	1:0	**Reserved** Format: MBZ
2	31:0	**Data DWord** Format: U32 FormatDesc This field specifies the DWord value to be written to the targeted location.

1.2.9 MI_NOOP

The MI_NOOP command format is:

MI_NOOP	
Project:	All
Source:	VideoCS
Length Bias:	1

The MI_NOOP command basically performs a “no operation” in the command stream and is typically used to pad the command stream (e.g., in order to pad out a batch buffer to a QWord boundary). However, there is one minor (optional) function this command can perform – a 22-bit value can be loaded into the MI NOPID register. This provides a general-purpose command stream tagging ("breadcrumb") mechanism (e.g., to provide sequencing information for a subsequent breakpoint interrupt).

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>0h MI_COMMAND</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:23</td><td><b>MI Command Opcode</b><table><tr><td>Default Value:</td><td>00h MI_NOOP</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>22</td><td><b>Identification Number Register Write Enable</b><table><tr><td>Format:</td><td>Enable</td></tr></table>This field enables the value in the Identification Number field to be written into the MI NOPID register. If disabled, that register is unmodified – making this command an effective “no operation” function.<table><tr><th>Value</th><th>Name</th></tr><tr><td>1</td><td>Write the NOP_ID register.</td></tr></table></td></tr>
<tr><td></td><td>21:0</td><td><b>Identification Number</b><table><tr><td>Format:</td><td>U22</td></tr></table>This field contains a 22-bit number which can be written to the MI NOPID register.</td></tr>
</table>

1.2.10 MI_SEMAPHORE_MBOX

MI_SEMAPHORE_MBOX

Source:	VideoCS
Length Bias:	2

Description	Project
This command is provided as alternative to MI_SEMAPHORE to provide mailbox-type semaphores where there is no update of the semaphore by the checking process (the consumer). Single-bit compare-and-update semantics are also provided. In either case, atomic access of semaphores need not be guaranteed by hardware as with the previous command. This command should eventually supersede the previous command. Synchronization between contexts (especially between contexts running on 2 different engines) is provided by the MI_SEMAPHORE_MBOX command. Note that contexts attempting to synchronize in this fashion must be able to access a common memory location. This means the contexts must share the same virtual address space (have the same page directory), must have a common physical page mapped into both of their respective address spaces, or the semaphore commands must be executing from a secure batch buffer or directly from a ring with the Use Global GTT bit set such that they are “privileged” and will use the (always shared) global GTT.MI_SEMAPHORE with the Update Semaphore bit set (and the Compare Semaphore bit clear) implements the Signal command, while the Wait command is indicated by Compare Semaphore being set. Note that Wait can cause a context switch. Signal increments unconditionally.	
If execution is stalled due to this command, the engine will specify that the engine is IDLE to the power management engine.	

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND
	28:23	**MI Command Opcode** Default Value: 16h MI_SEMAPHORE_MBOX
	22	**Use Global GTT** Format: U1 If set, this command will use the global GTT to translate the Semaphore Address and this command must be executing from a privileged (secure) batch buffer. If clear, the PPGTT will be used to translate the Semaphore Address. This bit will be ignored (and treated as if clear) if this command is executed from a non-privileged batch buffer. It is allowed for this bit to be clear when executing this command from a privileged (secure) batch buffer or directly from a ring buffer. **Programming Notes** This field is only valid when Compare Register Field is reset.
	21	**Update Semaphore** Format: U1 If set, the value from the Semaphore Data Dword is written to memory. If Compare Semaphore is also set, the semaphore is not updated if the semaphore comparison fails. If clear, the data at Semaphore Address is not changed. **Programming Notes** This field should be always clear when Compare Register Field is set.
	20	**Compare Semaphore** Format: U1 If set, the value from the **Semaphore Data Dword** is compared to the value from the **Semaphore Address** in memory when Compare Register is clear. If set, the value from the **Semaphore Data**

<table>
<tr><th colspan="3">MI_SEMAPHORE_MBOX</th></tr>
<tr><td></td><td></td><td><b>Dword</b> is compared to the value from <b>MMIO Register</b> selected by <b>Register Select</b> field when Compare Register is set. If the value at <b>Semaphore Address/MMIO Register is greater than the Semaphore Data Dword</b>, execution is continued from the current command buffer. If clear, no comparison takes place. <b>Update Semaphore</b><i>must</i> be set in this case.</td></tr>
<tr><td></td><td>19</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>18</td><td><b>Compare Register</b><table><tr><td></td><td></td></tr><tr><td>Format:</td><td>Compare Type</td></tr></table>If set, data in MMIO register will be used for compare. If clear, data in memory will be used for compare.<table><tr><th>Programming Notes</th></tr><tr><td>Compare Register field should be always set.</td></tr></table></td></tr>
<tr><td></td><td>17:16</td><td><b>Register Select</b><table><tr><td></td><td></td></tr><tr><td>Format:</td><td>Register Select</td></tr></table>If compare register is set in bit[18], this field indicates which register will be used.<table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>BCS register (VBSYNC)</td></tr><tr><td>2</td><td>CS register (VRSYNC)</td></tr><tr><td>3</td><td>Reserved</td></tr></table></td></tr>
<tr><td></td><td>15:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>0h Excludes DWord (0,1)</td></tr><tr><td>Format:</td><td>=n Total Length - 2</td></tr></table></td></tr>
<tr><td>1</td><td>31:0</td><td><b>Semaphore Data Dword</b><table><tr><td>Format:</td><td>U32</td></tr></table>Data dword to compare/update memory. The Data dword is supplied by software to control execution of the command buffer. If the compare is enabled and the data at Semaphore Address is greater than this dword, the execution of the command buffer continues.</td></tr>
<tr><td>2</td><td>31:2</td><td><b>PointerBitFieldName/MMIO Register Address</b><table><tr><td>Format:</td><td>GraphicsVirtualAddress[31:2]Semaphore</td></tr></table>if Compare Register bit[18] is cleared, this field if the Graphics Memory Address of the 32 bit value for the semaphore. If Compare Register bit[18] is set, this field is the MMIO address of the register for the semaphore.</td></tr>
<tr><td></td><td>1:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
</table>

1.2.11 MI_STORE_REGISTER_MEM

<table>
<tr><th colspan="3">MI_STORE_REGISTER_MEM</th></tr>
<tr><td colspan="3">
<table>
<tr><td>Project:</td><td>All</td></tr>
<tr><td>Source:</td><td>VideoCS</td></tr>
<tr><td>Length Bias:</td><td>2</td></tr>
</table>
</td></tr>
<tr><td colspan="3">The MI_STORE_REGISTER_MEM command requests a register read from a specified memory mapped register location in the device and store of that DWord to memory. The register address is specified along with the command to perform the read.</td></tr>
<tr><td colspan="3">
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>
<ul>
<li>The command temporarily halts command execution.</li>
<li>The memory address for the write is snooped on the host bus.</li>
<li>This command will cause undefined data to be written to memory if given register addresses for the PGTBL_CTL_0 or FENCE registers</li>
</ul>
The following addresses should NOT be used for SRMs
<ol>
<li>0x8800 - 0x88FF</li>
<li>>= 0x40000</li>
</ol>
The only exception is an SRM cycle to 0x40000-0xBFFFF when used as part of the LRI read-after-write requirement.
</td></tr>
</table>
</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b>
<table><tr><td>Default Value:</td><td>0h MI_COMMAND</td></tr></table>
</td></tr>
<tr><td></td><td>28:23</td><td><b>MI Command Opcode</b>
<table><tr><td>Default Value:</td><td>24h MI_STORE_REGISTER_MEM</td></tr></table>
</td></tr>
<tr><td></td><td>22</td><td><b>Use Global GTT</b>
This bit must be ‘1’ if the Per Process GTT Enable bit is clear.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Per Process Graphics Address</td><td></td></tr>
<tr><td>1h</td><td>Global Graphics Address</td><td>This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.</td></tr>
</table>
<table>
<tr><th>Programming Notes</th><th>Project</th></tr>
<tr><td>This will not be ignored when in a PPGTT batch buffer.</td><td></td></tr>
</table>
</td></tr>
<tr><td></td><td>21:8</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table>
</td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b>
<table>
<tr><td>Default Value:</td><td>1h Excludes DWord (0,1)</td></tr>
<tr><td>Format:</td><td>=n Total Length - 2</td></tr>
</table>
</td></tr>
<tr><td>1</td><td>31:23</td><td><b>Reserved</b>
<table>
<tr><td></td><td></td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
</table>
</td></tr>
<tr><td></td><td>22:2</td><td><b>Register Address</b>
<table><tr><td>Format:</td><td>MMIOAddress[22:2]MMIO_Register</td></tr></table>
This field specifies Bits 22:2 of the Register offset the DWord will be read from. As the register address
</td></tr>
</table>

MI_STORE_REGISTER_MEM

		must be DWord-aligned, Bits 1:0 of that address MBZ. **Programming Notes** Storing a VGA register is not permitted and will store an UNDEFINED value. The values of PGTBL_CTL0 or any of the FENCE registers cannot be stored to memory; UNDEFINED values will be written to memory if the addresses of these registers are specified.
	1:0	**Reserved** Format: MBZ
2	31:2	**Memory Address** Format: GraphicsAddress[31:2]MMIO_Register This field specifies the address of the memory location where the register value specified in the DWord above will be written. The address specifies the DWord location of the data.Range = GraphicsVirtualAddress[31:2] for a DWord register
	1:0	**Reserved** Format: MBZ

1.2.12 MI_STORE_DATA_IMM

The MI_STORE_DATA_IMM command format is:

MI_STORE_DATA_IMM

Source:	VideoCS
Length Bias:	2

The MI_STORE_DATA_IMM command requests a write of the QWord or DWord constant supplied in the packet to the specified Memory Address. As the write targets a System Memory Address, the write operation is coherent with the CPU cache (i.e., the processor cache is snooped).

Programming Notes	Project
This command should not be used within a "non-secure" batch buffer to access global virtual space. Doing so will cause the command parser to perform the write with byte enables turned off. This command can be used within ring buffers and/or "secure" batch buffers.	
Use Global GTT will not be ignored when in a PPGTT batch buffer.	
This command can be used for general software synchronization through variables in cacheable memory (i.e., where software does not need to poll un-cached memory or device registers).	
This command simply initiates the write operation with command execution proceeding normally. Although the write operation is guaranteed to complete "eventually", there is no mechanism to synchronize command execution with the completion (or even initiation) of these operations.	

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 20h MI_STORE_DATA_IMM Format: OpCode
	22	**Use Global GTT**

MI_STORE_DATA_IMM

		Format: U32 If set, this command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer. If clear, the PPGTT will be used. It is allowed for this bit to be clear when executing this command from a privileged (secure) batch buffer. This bit must be '1' if the Per Process GTT Enable bit is clear.
	21:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) = 3 for QWord, 2 for DWord Format: =n Total Length - 2
1	31:0	**Reserved** Format: MBZ
2	31:2	**Address** Format: GraphicsAddress[31:2] This field specifies Bits 31:2 of the Address where the DWord will be stored. As the store address must be DWord-aligned, Bits 1:0 of that address MBZ. This address must be 8B aligned for a store "QW" command.
	1:0	**Reserved** Format: MBZ
3	31:0	**Data DWord 0** Format: U32 FormatDesc This field specifies the DWord value to be written to the targeted location.For a QWord write this DWord is the lower DWord of the QWord to be reported (DW 0).
4	31:0	**Data DWord 1** Format: U32 FormatDesc This field specifies the upper DWord value to be written to the targeted QWord location (DW 1).

1.2.13 MI_STORE_DATA_INDEX

The MI_STORE_DATA_INDEX command format is:

MI_STORE_DATA_INDEX

Project:	All
Source:	VideoCS
Length Bias:	2

The MI_STORE_DATA_INDEX command requests a write of the data constant supplied in the packet to the specified offset from the System Address defined by the Hardware Status Page Address Register. As the write targets a System Address, the write operation is coherent with the CPU cache (i.e., the processor cache is snooped).

Programming Notes

- Use of this command with an invalid or uninitialized value in the Hardware Status Page Address Register is UNDEFINED.
- This command can be used for general software synchronization through variables in cacheable memory (i.e., where software does not need to poll uncached memory or device registers).
- This command simply initiates the write operation with command execution proceeding normally. Although the write operation is guaranteed to complete “eventually”, there is no mechanism to synchronize command execution with the completion (or even initiation) of these operations.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 21h MI_STORE_DATA_INDEX Format: OpCode
	22	**Reserved** Format: MBZ
	21	**Reserved** Format: MBZ
	20:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) = 2 for QWord Format: =n Total Length - 2
1	31:12	**Reserved** Format: MBZ
	11:2	**Offset** Format: U10 FormatDesc; zero-based DWord offset into the HW status page Format: GraphicsAddress[31:0]U32 This field specifies the offset (into the hardware status page) to which the data will be written. Note that the first few DWords of this status page are reserved for special-purpose data storage – targeting these reserved locations via this command is UNDEFINED.For a QWord write, the offset is valid down to bit 3

<table>
<tr><th colspan="3">MI_STORE_DATA_INDEX</th></tr>
<tr><td></td><td></td><td>only.<table><tr><th>Value</th><th>Name</th></tr><tr><td>[16, 1023]</td><td></td></tr></table></td></tr>
<tr><td></td><td>1:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>2</td><td>31:0</td><td><b>Data DWord 0</b><table><tr><td>Format:</td><td>U32 FormatDesc</td></tr></table>This field specifies the upper DWord value to be written to the targeted QWord location (DW 1).</td></tr>
<tr><td>3</td><td>31:0</td><td><b>Data Word 1</b><table><tr><td>Format:</td><td>U32 FormatDesc</td></tr></table>This field specifies the upper DWord value to be written to the targeted QWord location (DW 1).</td></tr>
</table>

1.2.14 MI_SUSPEND_FLUSH

<table>
<tr><th colspan="3">MI_SUSPEND_FLUSH</th></tr>
<tr><td colspan="3">Project: All
Source: VideoCS
Length Bias: 1</td></tr>
<tr><td colspan="3"><table><tr><th>Description</th><th>Project</th></tr><tr><td>Blocks MMIO sync flush or any flushes related to VT-d while enabled.</td><td></td></tr></table></td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>0h MI_COMMAND</td></tr></table></td></tr>
<tr><td></td><td>28:23</td><td><b>MI Command Opcode</b><table><tr><td>Default Value:</td><td>0Bh MI_SUSPEND_FLUSH</td></tr></table></td></tr>
<tr><td></td><td>22:1</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>0</td><td><b>Suspend Flush</b><table><tr><td>Format:</td><td>Enable</td></tr></table><table><tr><th>Description</th><th>Project</th></tr><tr><td>This field suspends flush due and IOTLB invalidation.</td><td></td></tr></table></td></tr>
</table>

1.2.15 MI_USER_INTERRUPT

MI_USER_INTERRUPT

Project:	All
Source:	VideoCS
Length Bias:	1

The MI_USER_INTERRUPT command is used to generate a User Interrupt condition. The parser will continue parsing after processing this command. See User Interrupt.

DWord	Bit	Description	
0	31:29	**Command Type**	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	**MI Command Opcode**	
		Default Value:	02h MI_USER_INTERRUPT
		Format:	OpCode
	22:0	**Reserved**	
		Format:	MBZ

1.2.16 MI_UPDATE_GTT

1.2.16.1 MI_UPDATE_GTT

MI_UPDATE_GTT

Source:	VideoCS
Length Bias:	2

The MI_UPDATE_GTT command is used to update GTT page table entries in a coherent manner and at a predictable place in the command flow. An MI_FLUSH should be placed before this command, because work associated with preceding commands that are still in the pipeline may be referencing GTT entries that will be changed by its execution. The flush will also invalidate TLBs and read caches that may become invalid as a result of the changed GTT entries. MI_FLUSH is not required if it can be guaranteed that the pipeline is free of any work that relies on changing GTT entries (such as MI_UPDATE_GTT contained in a paging DMA buffer that is doing only update/mapping activities and no rendering). This is a privileged command.

DWord	Bit	Description		
0	31:29	**Command Type**		
		Default Value:	0h MI_COMMAND	
		Format:	OpCode	
	28:23	**MI Command Opcode**		
		Default Value:	23h MI_UPDATE_GTT	
		Format:	OpCode	
	22	**Use Global GTT** Reserved: Must be 1h. Updating Per Process Graphics Address is not supported		
		Value	Name	Description
		0h	Per Process Graphics Address	
		1h	Global Graphics Address	

MI_UPDATE_GTT

	21:6	**Reserved** Format: MBZ
	5:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:12	**Entry Address** Format: GraphicsAddress[31:12] This field simply holds the DW offset of the first table entry to be modified. Note that one or more of the upper bits may need to be 0, i.e., for a 2G aperture, bit 31 MBZ.
	11:0	**Reserved** Format: MBZ
2..n	31:0	**Entry Data** Format: Page Table Entry This Dword becomes the new page table entry. See PPGTT/Global GTT Table Entries (PTEs) in Memory Interface Registers.

1.2.17 MI_WAIT_FOR_EVENT

MI_WAIT_FOR_EVENT

Source:	VideoCS
Length Bias:	1

The MI_WAIT_FOR_EVENT command is used to pause command stream processing of this pipe only until a specific event occurs or while a specific condition exists. See Wait Events/Conditions, Device Programming Interface in MI Functions. Only one event/condition can be specified -- specifying multiple events is UNDEFINED.Note that if a specified condition does not exist (the condition code is inactive) at the time the parser executes this command, the parser proceeds, treating this command as a no-operation.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND
	28:23	**MI Command Opcode** Default Value: 03h MI_WAIT_FOR_EVENT
	22:20	**Reserved** Project: All Format: MBZ
	19:16	**Condition Code Wait Select** This field enables a wait for the duration that the corresponding condition code is active. These enable select one of 15 condition codes in the EXCC register, that cause the parser to wait until that condition-code in the EXCC is cleared.

MI_WAIT_FOR_EVENT

Value	Name	Description
0h	Not enabled	Condition Code Wait Not Enabled
1h-5h	Enable	Condition Code select enabled; selects one of 5 codes, 0 – 4
6h-15h	Reserved	

Programming Notes
Note that not all condition codes are implemented. The parser operation is UNDEFINED if an unimplemented condition code is selected by this field. The description of the EXCC register (Memory Interface Registers) lists the codes that are implemented.

15:0 **Reserved**

Format:	MBZ

1.2.18 MI_LOAD_REGISTER_MEM

MI_LOAD_REGISTER_MEM

Source:	VideoCS
Length Bias:	2

The MI_LOAD_REGISTER_MEM command requests from a memory location and stores that DWord to a register.

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

Default Value:	0h MI_COMMAND
Format:	OpCode

DWord	Bit	Description
	28:23	**MI Command Opcode**

Default Value:	29h MI_LOAD_REGISTER_MEM
Format:	OpCode

DWord	Bit	Description
	22	**Use Global GTT** This bit must be 1 if the Per-Process GTT Enable bit is clear.

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

DWord	Bit	Description
	21	**Async Mode Enable** If this bit is set then the command stream will not wait for completion of this command before executing the next command.
	20:8	**Reserved**

Format:	MBZ

DWord	Bit	Description
	7:0	**DWord Length**

Format:	=n

Value	Name	Description
01h	Excludes DWord (0,1) **[Default]**	Total Length - 2

MI_LOAD_REGISTER_MEM		
1	31:26	**Reserved** Format: MBZ
	22:2	**Register Address** Format: MMIOAddress[22:2]MMIO_Register This field specifies Bits 25:2 of the Register offset the DWord will be written to. As the register address must be DWord-aligned, Bits 1:0 of that address MBZ.
	1:0	**Reserved** Format: MBZ
2	31:2	**Memory Address** Format: GraphicsAddress[31:2]MMIO_Register This field specifies the address of the memory location where the register value specified in the DWord above will read from. The address specifies the DWord location of the data.
	1:0	**Reserved** Format: MBZ

Revision History

Revision Number	Description	Revision Date
1.0	First 2012 OpenSource edition	May 2012

§§