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

Volume 1 Part 4: Graphics Core – Video Codec Engine (SandyBridge)

For the 2011 Intel Core Processor Family

May 2011

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.

Doc Ref #: IHD-OS-V1 Pt4 – 05 11

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. Intel products are not intended for use in medical, life saving, or life sustaining applications.

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 SandyBridge chipset family, Havendale/Auburndale chipset family, Intel® 965 Express Chipset Family, Intel® G35 Express Chipset, and Intel® 965GMx Chipset Mobile Family Graphics Controller 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.

Contact your local Intel® sales office or your distributor to obtain the latest specifications and before placing your product order. I2C is a two-wire communications bus/protocol developed by Philips. SMBus is a subset of the I2C bus/protocol and was developed by Intel®. 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 4

1.1 Registers for Video Codec 4
1.1.1 Introduction 4
1.1.2 Virtual Memory Control 4
1.1.3 Context Submission [DevSNB] 11
1.1.4 VCS_RINGBUF—Ring Buffer Registers 13
1.1.5 Watchdog Timer Registers 16
1.1.6 Interrupt Control Registers 17
1.1.7 Logical Context Support 24
1.1.8 Registers in MFC Pipe [DevSNB+] 26
1.1.9 Registers in Media Engine 33
1.2 Memory Interface Commands for Video Codec Engine 35
1.2.1 Introduction 35
1.2.2 MI_ARB_CHECK 35
1.2.3 MI_ARB_ON_OFF 36
1.2.4 MI_BATCH_BUFFER_END 37
1.2.5 MI_CONDITIONAL_BATCH_BUFFER_END 37
1.2.6 MI_BATCH_BUFFER_START 39
1.2.7 MI_FLUSH_DW 41
1.2.8 MI_LOAD_REGISTER_IMM 44
1.2.9 MI_NOOP 45
1.2.10 MI_REPORT_HEAD 46
1.2.11 MI_SEMAPHORE_MBOX 47
1.2.12 MI_STORE_REGISTER_MEM 49
1.2.13 MI_STORE_DATA_IMM 51
1.2.14 MI_STORE_DATA_INDEX 53
1.2.15 MI_SUSPEND_FLUSH 55
1.2.16 MI_USER_INTERRUPT 55
1.2.17 MI_UPDATE_GTT 56
1.2.18 MI_WAIT_FOR_EVENT 57

1. Video Codec Engine Command Streamer

For [DevSNB+], 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
DevSNB+	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_DIR_BASE – Page Directory Base Register

VCS_PP_DIR_BASE – Page Directory Base Register	
Register Type:	MMIO_CS
Address Offset:	12228h
Project:	All
Default Value:	0000 0000h
Access:	R/W
Size (in bits):	32

This register contains the offset into the GGTT where the (current context's) PPGTT page directory begins. This register is restored with context. The Page Directory Base Address is set by SW only by modifying the value of this register in the context image such that the new value is restored the next time the context runs. A write via MMIO to this register triggers the render pipe to fetch all PDs.

Programming Note: The **MBC Driver Boot Enable** bit in **MBCTL** register must be set *before* this register is written to upon boot up (including S3 exit)

Bit	Description
30:16	**Page Directory Base Offset** Project: All Default Value: 0h Format: U15 Range [0,GGTT Size in cachelines - 1] Contains the cacheline (64-byte) offset into the GGTT where the page directory begins.
15:1	**Reserved** Project: All Format: MBZ
0	**PD Load Busy** Project: DevSNB+ Format: Valid This is a read-only field that indicates if the page directories are currently being fetched and loaded.

1.1.2.2 VCS_PP_DCLV – PPGTT Directory Cacheline Valid Register

VCS_PP_DCLV – PPGTT Directory Cacheline Valid Register	
Register Type:	MMIO_CS
Address Offset:	12220h
Project:	All
Default Value:	0h
Access:	R/W
Size (in bits):	64

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 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

Bit	Description
63:32	**Reserved** Project: All Format: MBZ
31:0	**PPGTT Directory Cache Restore [1..32] 16 entries** Project: All Format: Array:Enable 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.

The field below needs to go in some register to enable PPGTT, either in GAC MMIO or VCS MMIO.

1	**Per-Process GTT Enable** Project: DevSNB+ Format: Enable If set, PPGTT support in hardware is enabled. This bit *must* be set if runlist enable is set. Setting this bit also allows support for big pages (32k)

1.1.2.3 VCS_MI_MODE — Mode Register for Software Interface

VCS_MI_MODE — Mode Register for Software Interface

Register Type:

Address Offset: 1209Ch–1209Fh

Project:

Default Value: 0000 0200h

Access: Read/Write

Size (in bits): 32 bits

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

Bit	Description
31:16	**Masks:** A “1” in a bit in this field allows the modification of the corresponding bit in Bits 15:0
15	**Suspend Flush** **Project:** **All** **Mask:** MMIO(0x209c)#31 **Value** / **Name** / **Description** / **Project** **Oh** / No Delay / HW will not delay flush, this bit will get cleared by MI_SUSPEND_FLUSH as well / **All** **1h** / Delay Flush / HW will delay the flush because of sync flush or VTD regimes until reset, this bit will get set by MI_SUSPEND_FLUSH as well / **All**
14:12	**Reserved** Read/Write
11	**Invalidate UHPTR enable:** If bit set H/W clears the valid bit of BCS_UHPTR (4134h, bit 0) when current active head pointer is equal to UHPTR.
10	**Reserved** Read/Write
9	**Ring Idle (Read Only Status bit)** 0 = Parser not Idle 1 = Parser Idle ***Writes to this bit are not allowed.***
8	**Stop Ring** 0 = Normal Operation. 1 = Parser is turned off. 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. ***Software must clear this bit for Ring to resume normal operation.***
7:3	**Reserved** Read/Write

VCS_MI_MODE — Mode Register for Software Interface	
Register Type:	
Address Offset:	1209Ch–1209Fh
Project:	
Default Value:	0000 0200h
Access:	Read/Write
Size (in bits):	32 bits
The MI_MODE register contains information that controls software interface aspects of the command parser	

Bit	Description
2	**MI_ARB_ON_OFF Privileged Attribute Enable [DevSNB+]** If set, the MI_ARB_ON_OFF command will be treated as a privileged command. That is, if executed in a non-secure batch buffer, hardware will convert it to a NOOP. If clear, hardware will execute at all times. Note that this register cannot be changed in the UMD.
1:0	**Reserved** Read/Write

1.1.2.4 VCS_INSTPM—Instruction Parser Mode Register

Address Offset: 120C0h–120C3h

Default Value: 0000 0000h

Access: Read/Write

Size: 32 bits

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.

Programming Notes:

- All Reserved bits are implemented.

Bit	Description
31:16	**Masks:** 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.
15:10	Reserved: MBZ
9	**Reserved** Project: Format:

Bit	Description
8:7	Reserved: MBZ
6	**Memory Sync Enable:** This set, this bit allows the video decode engine to write out the data from the local caches to memory. **[DevSNB+]** This bit is *not* persistent. S/W must define this bit each time a sync flush is requested
5	**Sync Flush Enable:** 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*). **Programming Note:** • 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**. Format = Enable (cleared by HW) [DevSNB+] When using MI_SUSPEND_FLUSH, this bit cannot be relied on as an indicator of sync flush complete. Instead, driver must wait until head == tail
4:0	Reserved: MBZ

1.1.2.5 VCS_NOPID — NOP Identification Register

Address Offset: 12094h–12097h

Default Value: 0000 0000h

Access: Read Only

Size: 32 bits

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

Bit	Description
31:22	Reserved: 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.

1.1.2.6 VBSYNC – Video/Blitter Semaphore Sync Register

VBSYNC – Video/Blitter Semaphore Sync Register	
Register Type:	MMIO_VCS
Address Offset:	12040h
Project:	All
Default Value:	00000000h
Access:	R/W
Size (in bits):	32
Trusted Type:	1
This register is written by BCS, read by VCS.	

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

1.1.2.7 VRSYNC – Video/Render Semaphore Sync Register

VRSYNC – Video/Render Semaphore Sync Register	
Register Type:	MMIO_VCS
Address Offset:	12044h
Project:	All
Default Value:	00000000h
Access:	R/W
Size (in bits):	32
Trusted Type:	1
This register is written by CS, read by VCS.	

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

1.1.2.8 GAC_MODE — Mode Register for GAC

Address Offset: 120A0h–120A3h

Default Value: 0000 0000h

Access: Read/Write

Size: 32 bits

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

Bit	Description
31:16	**Masks:** A “1” in a bit in this field allows the modification of the corresponding bit in Bits 15:0
15:0	**Reserved** Read/Write

1.1.3 Context Submission [DevSNB]

1.1.3.1 VCS_RCCID—Ring Buffer Current Context ID Register

Address Offset: 12190h–12197h

Default Value: 00 00 00 00h

Access: Read/Write

Size: 32 bits

This register contains the current “ring context ID” associated with the ring buffer.

Programming Notes:

- The current context registers must not be written directly (via MMIO). The RCCID register should only be updated indirectly from RNCID.

Bit	Description
63:0	See Context Descriptor for VCS

1.1.3.2 VCS_RNCID—Ring Buffer Next Context ID Register

Address Offset: 12198h–1219fh

Default Value: 00 00 00 00h

Access: Read/Write

Size: 64 bits

This register contains the *next* “ring context ID” associated with the ring buffer.

Programming Notes:

- The current context (RCCID) register can be updated indirectly from this register on a context switch event. Note that this can only be triggered when arbitration is enabled or if the current context runs dry (head pointer becomes equal to tail pointer).

Bit	Description
63:0	See Context Descriptor for VCS

1.1.3.3 Context Status

A context switch interrupt will be sent anytime a context switch occurs. This is documented in the "GPU Overview" volume, "Memory Data Formats" chapter. A status DW for the context that was just switched away from will be written to the Context Status Buffer in the Global Hardware Status Page. The status contains the context ID and the reason for the context switch. Note that since there will have been no running contexts when the very first (after reset) context is submitted, the Context ID in the first Context Status DWord will be UNDEFINED.

Table 1-1. Format of Context Status Dword

Bit	Description
31:12	**Context ID.** Contains the context ID copied from the submitted context.
11:8	Reserved: MBZ
7	**Media watch dog timer expired** cause the context switch
6	Reserved: MBZ
5	Reserved: MBZ
4	**Ring Buffer Becoming Empty** Caused context to Switch.
3	Reserved: MBZ
2	Reserved: MBZ
1	**Waiting on a Semaphore** Caused Context to Switch.
0	Reserved: MBZ

When SW services a context switch interrupt, it should read the Context Status Buffer beginning where it left off reading the last time it serviced a context switch interrupt. It should read up through the **Last Written Status Offset**, which is also recorded in the Context Status Buffer. The status DWs can be examined to determine which contexts were switched out between context interrupt service intervals, and why.

Table 1-2. Number of Context Status Entries in Memory

Device	Number of Status Entries
DevSNB	12 (DW) Entries

Status Dwords are written out to the Context Status Buffer at incrementing addresses. The Context Status Buffer has a limited size and simply wraps around to the beginning when the end is reached.

The Context Status Buffer fits into a single cacheline so that the whole buffer will be read from memory at once if the driver performs a cacheable read.

Table 1-3. Format of the Context Status Buffer

DW	Description
15	**Last Written Status Offset.** This Dword is written on every context switch with the (pre-increment) value of the **Context Status Buffer Pointer Register**. The lower 4 bits increment for every status Dword write; the upper 28 bits are always 0. The lowest 4 bits indicate which of the Context Status Dwords was just written.
14-12	Reserved: MBZ
11-0	**Context Status Dwords.** A circular buffer of context status DWs. As each context is switched away from, its status is written here at ascending DWs as indicated by the **Last Written Status Offset**. Once DW 11 has been written, the pointer wraps around so that the next status will be written at DW0. Format = ContextStatusDW

1.1.4 VCS_RINGBUF—Ring Buffer Registers

Address Offset: 12030h – 1203Fh: Ring Buffer:

offset 0h = _TAIL

offset 4h = _HEAD

offset 8h = _START

offset Ch = _CTL

Default Value: 0000 0000h

Access: Read/32 bit Write Only

Size: 4 DWords / Ring Buffer

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

The format of the Ring Buffer register set follows:

<table>
<tr><th>DWord Offset</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="3">0</td><td>31:21</td><td>Reserved: MBZ</td></tr>
<tr><td>20:3</td><td>Tail Offset: 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.
Format = U18 QWord Offset

[DevSNB] Every tail move must follow the sequence below
<table>
<tr><th>MMIO action</th><th>Comment</th></tr>
<tr><td>Write 0x12050 = 0x00010001</td><td>Disable MFX pipe from claiming idle</td></tr>
<tr><td>Write 0x12198 = 0x00000000</td><td>Benign active cycle that will wake up MFX pipe (if currently idle)</td></tr>
<tr><td>Poll for 0x12050[3] = 0</td><td>Make sure MFX pipe is out of idle. Very unlikely will need more than 1 read</td></tr>
<tr><td>Write 0x12030 = <new tail ptr value></td><td></td></tr>
<tr><td>Write 0x12050 = 0x00010000</td><td>Let VCS claim MFX pipe idle again</td></tr>
</table></td></tr>
<tr><td>2:0</td><td>Reserved: MBZ</td></tr>
<tr><td rowspan="2">1</td><td>31:21</td><td>Wrap Count: 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.
Format = U11 count of ring buffer wraps</td></tr>
<tr><td>20:2</td><td>Head 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.

Format = U19 DWord Offset</td></tr>
<tr><td></td><td>1:0</td><td>Reserved: MBZ</td></tr>
</table>

DWord Offset	Bit	Description
2	31:12	**Starting Address:** This field specifies Bits 31:12 of the 4KB-aligned starting Graphics Address of the ring buffer. All ring buffer pages must map to Main Memory (uncached) pages. Ring Buffer addresses are always translated through the global GTT. Per-process address space can only be used via a batch buffer. Format: Graphics Address Bits 31:12
	11:0	Reserved: MBZ
3	31:21	Reserved: MBZ
	20:12	**Buffer Length:** This field is written by SW to specify the length of the ring buffer in 4 KB Pages. Format = U9 in 4 KB pages – 1 Range = [0 = 1 page = 4 KB, 1FFh = 512 pages = 2 MB]
	11	**RBWait** 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.
	10	**Semaphore Wait** Indicates that this ring has executed a MI_SEMAPHORE_MBOX instruction with register compare 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 the compare to meet and this bit is cleared, the wait will be terminated and the RB will be returned to arbitration.
	9	**Reserved:** MBZ
	8	**Disable Register Accesses:** 0 = Ring is allowed to access (read or write) MMIO space. 1 = Ring is not allowed to write MMIO space. Ring *is* allowed to read registers.
	7:3	Reserved: MBZ
	2:1	**Automatic Report Head Pointer:** 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. Format = 0: MI_AUTOREPORT_OFF – Automatic reporting disabled 1: MI_AUTOREPORT_64KB – Report every 16 pages (64KB) 2: MI_AUTOREPORT_4KB – Report every page (4KB) 3: MI_AUTOREPORT_128KB – Report every 32 pages (128KB) When the **Per-Process Virtual Address Space** bit is set and automatic head reporting is desired, this field must be set to option 2 since the ring buffer will be only 16KB in size. The head pointer will be reported to the head pointer location in the PP HW 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.
	0	**Ring Buffer 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. Format = Enable

1.1.4.1 VCS_UHPTR — Pending Head Pointer Register

Address Offset: 12134h–12137h

Default Value: 0000 0000h

Access: Read/Write

Size: 32 bits

Bit	Description
31:3	**Head Pointer Address**: This register represents the GFX address offset where execution should continue in the ring buffer following execution of an MI_ARB_CHECK command. Format = MI_Graphics_Offset
2:1	Reserved: MBZ
0	**Head Pointer Valid:** 1 = Indicates that there is an updated head pointer programmed in this register 0 = No valid updated head pointer register, resume execution at the current location in the ring buffer 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.

1.1.5 Watchdog Timer Registers

1.1.5.1 VCS_CNTR—Counter for the bit stream decode engine

Address Offset: 12178h–1217Bh

Default Value: FFFF FFFFh

Access: Read/Write

Size: 32 bits

Bit	Description
31:0	**Count Value**: 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—Threshold for the counter of bit stream decode engine

Address Offset: 1217Ch–1217Fh

Default Value: 00150000h

Access: Read/Write

Size: 32 bits

Bit	Description
31:0	**Threshold Value**: 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:

Table 1-4. 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. **Per-Process Virtual Address Space bit** needs to be set for this interrupt to occur**.**
19	**Page Fault:** This bit is set whenever there is a pending PPGTT (page or directory) fault.
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 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

Bit	Description
13	Reserved
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
1	Reserved	
0	User Interrupt	0

1.1.6.1 HWSTAM — Hardware Status Mask Register

Hardware Status Mask Register	
Register Type:	MMIO_VCS
Address Offset:	12098h
Project:	All
Default Value:	**FFFF FFFFh**
Access:	R/W, RO for Reserved Control bits
Size (in bits):	32
Trusted Type:	1

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.

Bit	Description
31:0	**Hardware Status Mask Register** Project: All Default Value: FFFFFFFFh DefaultVaueDesc Format: Array of Masks refer to Table 4-4 in Interrupt Control Register section for bit definitions

1.1.6.2 IMR—Interrupt Mask Register

IMR—Interrupt Mask Register

Register Type: MMIO_VCS
Address Offset: 120A8h
Project: All
Default Value: FFFF FFFFh
Access: R/W
Size (in bits): 32

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.

Bit	Description
31:0	**Interrupt Mask Bits** Project: All Default Value: FFFF FFFFh Format: Array of interrupt mask bits — Refer to 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	Project
0h	Not Masked	Will be reported in the IIR	All
1h	Masked	Will not be reported in the IIR	All

1.1.6.3 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:

Table 1-5. Hardware-Detected Error Bits

Bit	Description
15:1	Reserved: MBZ
0	**Instruction Error:** This bit is set when the Renderer Instruction Parser detects an error while parsing an instruction. Instruction errors include: 1) Client ID value (Bits 31:29 of the Header) is not supported (only MI, 2D and 3D are supported). 2) Defeatured MI Instruction Opcodes: 1: Instruction Error detected **Programming Note:** This error indications can not be cleared except by reset (i.e., it is a fatal error).

1.1.6.3.1 EIR — Error Identity Register

EIR — Error Identity Register						
Register Type:	MMIO_VCS					
Address Offset:	120B0h					
Project:	All					
Default Value:	0000 0000h					
Access:	R/WC					
Size (in bits):	32					
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).						
Bit	**Description**					
31:16	**Reserved**	Project:	All	Format:	MBZ	

EIR — Error Identity Register

15:0	**Error Identity Bits** Project: All Default Value: 0h Format: Array of Error condition bits See Table 1 5. Hardware-Detected Error Bits This register contains the persistent values of ESR error status bits that are unmasked via the EMR register. (See **Error! Reference source not found.**). 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	Project
1h	Error occurred	Error occurred	All

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

1.1.6.3.2 EMR—Error Mask Register

EMR—Error Mask Register

Register Type: MMIO_VCS
Address Offset: 120B4h
Project: All
Default Value: FFFF FFFFh
Access: R/W
Size (in bits): 32

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.

Bit	Description
31:16	**Reserved** Project: All Format: MBZ
15:0	**Error Mask Bits** Project: All Default Value: FFFF FFFFh Format: Array of error condition mask bits See Table 1 5. 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	Project
0h	Not Masked	Will be reported in the EIR	All
1h	Masked	Will not be reported in the EIR	All

1.1.6.3.3 ESR—Error Status Register

ESR—Error Status Register

Register Type: MMIO_VCS
Address Offset: 120B8h
Project: All
Default Value: 0000 0000h
Access: RO
Size (in bits): 32

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.

Bit	Description
31:16	**Reserved** Project: All Format: MBZ

ESR—Error Status Register	
15:0	**Error Status Bits** Project: All Default Value: 0h Format: Array of error condition bits See Table 1 5. Hardware-Detected Error Bits This register contains the non-persistent values of all hardware-detected error condition bits.

Value	Name	Description	Project
1h	Error Condition Detected	Error Condition detected	All

1.1.7 Logical Context Support

1.1.7.1 VCS_BB_ADDR—Batch Buffer Head Pointer Register

Address Offset: 012140h–012147h

Default Value: 0000 0000 0000 0000h

Access: Read-Only

Size: 64 bits

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

Bit	Description
63:32	Reserved: MBZ
31:3	**Batch Buffer Head Pointer:** This field specifies the QWord-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. .
2:1	Reserved: MBZ
0	**Valid:** 1 = Batch buffer Valid 0 = Batch buffer Invalid

1.1.7.2 VCS_BB_STATE — Batch Buffer State Register

VCS_BB_STATE – Batch Buffer State Register

Register Type:	MMIO_VCS
Address Offset:	12110h
Project:	All
Default Value:	0000 0000h
Access:	R/W
Size (in bits):	32

This register contains the attributes of the last 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.

Bit	Description
31:6	**Reserved** Project: All Format: MBZ
5	**Buffer Security Indicator** Project: All Default Value: 0h Format: MI_BufferSecurityType If set, this batch buffer is non-secure and cannot execute privileged commands nor access privileged (GGTT) memory. It will be accessed via the PPGTT. If clear, this batch buffer is secure and will be accessed via the GGTT. Note: This field reflects the effective security level and may not be the same as the Buffer Security Indicator written using MI_BATCH_BUFFER_START. **Value** / **Name** / **Description** / **Project** 0h / MIBUFFER_SECURE / Located in GGTT memory / All 1h / MIBUFFER_NONSECURE / Located in PPGTT memory / All
4	**Reserved**
3:0	**Reserved** Project: All Format: MBZ

1.1.7.3 VCS_CTXT_SR_CTL — Context Save/Restore Control Register

CTXT_SR_CTL – Context Save/Restore Control Register	
Register Type:	MMIO_VCS
Address Offset:	12114h
Project:	All
Default Value:	0000 0000h
Access:	R/W
Size (in bits):	32
This register is saved and restored with context.	

Bit	Description
31:1	**Reserved** Project: Format: MBZ
0	**MFX Context Restore Inhibit** Project: Format: U1 iis is not a true register bit. This bit should be set in the context image of a ring context that is being submitted for the first time. Setting this bit will inhibit the restoring of render context (including extended context if applicable) so that restoring of an uninitialized render context can be prevented. This bit will always be set on a context save (since the render context cannot be uninitialized on context save – it will always contain at least default values.)

1.1.8 Registers in MFC Pipe [DevSNB+]

These registers count for AVC encoder statistics of the parallel Video Codec Engine (VCE). They are saved and restored with context but should not be changed by software during frame processing. These registers are reset to 0 each time when command MFX_PIPE_MODE_SELECT is issued. These registers may be read at any time; however, to obtain a meaningful result, a pipeline flush just prior to reading the registers is necessary in order to synchronize the counts with the primitive stream. These registers can be read to memory through the MI_STORE_REGISTER_MEM command.

1.1.8.1 MFC_VIN_AVD_ERROR_FLAGS — AVC Bitstream Decoding Front-End Parsing Logic Error Report Register

MFC_VIN_AVD_ERROR_FLAGS	
Register Type:	MMIO_VCS
Address Offset:	12400h
Project:	All
Default Value:	00000000h;
Access:	RW
Size (in bits):	32
Trusted Type:	1
The only MMIO write operation is to reset each individual bit of this register to a 0 value, after the error information has been read and/or processed. The driver may choose to read this register in between pictures and video sequence and upon video stream switching. This register is set to 0 at powerup.	

MFC_VIN_AVD_ERROR_FLAGS	
Bit	**Description**
31:0	avd_error_flagsR[31:0] 31:6 -- Reserved 5 – **AVD Error Rewind flag** 4 – **AVD Error Conceal Flag** 3 -- **BSD Premature Completion Error Status Flag** When a BSD Premature Completion error has occurred and the BSDPrematureComplete Error Handling bit in the inline data of the AVC_BSD_OBJECT command is set, this error status flag is set until being cleared by a subsequent MMIO write to this register. 2 -- **MPR Error Status Flag** When a MPR error has occurred and the MPR Error Handling bit in the inline data of the AVC_BSD_OBJECT command is set, this error status flag is set until being cleared by a subsequent MMIO write to this register. 1 -- **VLD Error Status Flag** When a VLD error has occurred and the VLD Error Handling bit in the inline data of the AVC_BSD_OBJECT command is set, this error status flag is set until being cleared by a subsequent MMIO write to this register. 0 -- **BSD Error Status Flag** When a BSD error has occurred and the BSD Error Handling bit in the inline data of the AVC_BSD_OBJECT command is set, this error status flag is set until being cleared by a subsequent MMIO write to this register.

1.1.8.2 MFC_VIN_AVD_ERROR_CNTR — AVC Bitstream Decoding Front-End Parsing Logic Error Counter Report Register

MFC_VIN_AVD_ERROR_CNTR[11:0]	
Register Type:	MMIO_VCS
Address Offset:	12404h
Project:	All
Default Value:	00000000h;
Access:	RW
Size (in bits):	32
Trusted Type:	1

The only MMIO write operation is to reset this register to a 0 value. The driver may choose to read this register in between pictures and video sequence and upon video stream switching. This register is set to 0 at powerup.

Bit	Description
31:0	**avd_error_flagsR[31:0]** : 31:12 -- Reserved 11:0 -- **BSD Error Count --** Increment by 1 when any of the recognized errors (BSD, VLD, MPR and PrematureCompletion) has occurred. Do not wrap around when the maximum count has reached. error_cntR[11:0]

1.1.8.3 MFC_BITSTREAM_BYTECOUNT_SLICE — Bitstream Output Byte Count Per Slice Report Register

MFC_BITSTREAM_BYTECOUNT_SLICE	
Register Type:	MMIO_VCS
Address Offset:	12408h
Project:	[DevSNB+]
Default Value:	00000000h; 00000000h;
Access:	RO
Size (in bits):	32
Trusted Type:	1
This register stores the count of bytes of the bitstream output. This register is part of the context save and restore.	

Bit	Description
31:0	**MFC Bitstream Byte Count** Total number of bytes in the bitstream output from the encoder. This count is updated for every time the internal bitstream counter is incremented.

1.1.8.4 MFC_BITSTREAM_SE_BITCOUNT_SLICE — Bitstream Output Bit Count for the last Syntax Element Report Register

MFC_BITSTREAM_SE_BITCOUNT_SLICE	
Register Type:	MMIO_VCS
Address Offset:	1240Ch
Project:	All
Default Value:	00000000h; 00000000h;
Access:	RO
Size (in bits):	32
Trusted Type:	1
This register stores the count of number of bits in the bitstream for the last syntax element before padding. The bit count is before the byte-aligned alignment padding insertion, but includes the stop-one-bit. This register is part of the context save and restore.	

Bit	Description
31:0	**FC Bitstream Syntax Element Bit Count** ıtal number of bits in the bitstream output before padding. This count is updated each time the internal counter is incremented.

1.1.8.5 MFC_AVC_CABAC_INSERTION_COUNT — Bitstream Output CABAC Insertion Count Report Register

MFC_AVC_CABAC_INSERTION_COUNT	
Register Type:	MMIO_VCS
Address Offset:	12410h
Project:	All
Default Value:	00000000h; 00000000h;
Access:	RO
Size (in bits):	32
Trusted Type:	1
This register stores the count in bytes of **CABAC ZERO_WORD insertion. It is primarily provided for statistical data gathering.** This register is part of the context save and restore.	
Bit	**Description**
31:0	**MFC AVC Cabac Insertion Count** Total number of bytes in the bitstream output before for the CABAC zero word insertion. This count is updated each time when the insertion count is incremented.

1.1.8.6 MFC_AVC_MINSIZE_PADDING_COUNT — Bitstream Output Minimal Size Padding Count Report Register

MFC_AVC_MINSIZE_PADDING_COUNT	
Register Type:	MMIO_VCS
Address Offset:	12414h
Project:	All
Default Value:	00000000h; 00000000h;
Access:	RO
Size (in bits):	32
Trusted Type:	1
This register stores the count in bytes of **minimal size padding insertion. It is primarily provided for statistical data gathering.** This register is part of the context save and restore.	
Bit	**Description**
31:0	**MFC AVC MinSize Padding Count** Total number of bytes in the bitstream output contributing to minimal size padding operation. This count is updated each time when the padding count is incremented.

1.1.8.7 MFC_IMAGE_STATUS_MASK

MFC_IMAGE_STATUS_MASK	
Register Type:	MMIO
Address Offset:	**12418H**
Project:	DevSNB+
Default Value:	00000000h; 00000000h;
Access:	RO
Size (in bits):	32
Trusted Type:	1
This register stores the image status(flags). This register is part of the context save and restore.	

Bit	Description
31:0	Control Mask for dynamic frame repeat

1.1.8.8 MFC_IMAGE_STATUS_CONTROL

MFC_IMAGE_STATUS_CONTROL	
Register Type:	MMIO
Address Offset:	**1241CH**
Project:	DevSNB+
Default Value:	00000000h; 00000000h;
Access:	RO
Size (in bits):	32
Trusted Type:	1
This register stores the suggested data for next frame in multipass. This register is part of the context save and restore.	

Bit	Description
31:24	**Reserved**
23:16	suggested slice QP delta value for frame level Rate control. This value can be +ve or -ve
15:2	**Reserved**
1	**Frame Bit count over-run/under-run flag**
0	Max Macroblock conformance flag or Frame Bit count over-run/under-run

1.1.8.9 MFC_BITSTREAM_BYTECOUNT_FRAME — Reported Bitstream Output Byte Count per Frame Register

BITSTREAM_BYTECOUNT_FRAME	
Register Type:	MMIO
Address Offset:	**12420H**
Project:	DevSNB+
Default Value:	00000000h; 00000000h;
Access:	RO
Size (in bits):	32
Trusted Type:	1
This register stores the count of bytes of the bitstream output per frame. This register is part of the context save and restore.	
Bit	**Description**
31:0	**MFC Bitstream Byte Count per Frame** Total number of bytes in the bitstream output per frame from the encoder. This includes header/tail/byte alignment/data bytes/EMU bytes/cabac-zero word insertion/padding insertion. This count is updated for every time the internal bitstream counter is incremented and its reset at image start.

1.1.8.10 MFC_BITSTREAM_SE_BITCOUNT_FRAME — Reported Bitstream Output Bit Count for Syntax Elements Only Register

MFC_BITSTREAM_SE_BITCOUNT_FRAME	
Register Type:	MMIO
Address Offset:	**12424H**
Project:	All
Default Value:	00000000h; 00000000h;
Access:	RO
Size (in bits):	32
Trusted Type:	1
This register stores the count of number of bits in the bitstream due to syntax elements only. This excludes header/ byte alignment /tail/EMU/CABAC-0word/padding bits but includes the stop-one-bit. This register is part of the context save and restore.	
Bit	**Description**
31:0	**MFC Bitstream Syntax Element Only Bit Count** Total number of bits in the bitstream output due to syntax elements only. It includes the data bytes only. This count is updated for every time the internal bitstream counter is incremented and its reset at image start.

1.1.8.11 MFC_AVC_CABAC_BIN_COUNT_FRAME — Reported Bitstream Output CABAC Bin Count Register

MFC_AVC_CABAC_BIN_COUNT_FRAME	
Register Type:	MMIO
Address Offset:	**12428H**
Project:	All
Default Value:	00000000h; 00000000h;
Access:	RO
Size (in bits):	32
Trusted Type:	1
This register stores the count of number of bins per frame. This register is part of the context save and restore.	

Bit	Description
31:0	**MFC AVC Cabac Bin Count** Total number of bins in the bitstream output per frame from the encoder. This count is updated for every time the bin counter is incremented and its reset at image start.

1.1.9 Registers in Media Engine

1.1.9.1 Introduction

The register detailed in this chapter is used across the GEN family of products and is an extention to previous projects. However, slight changes may be present (i.e., for features added or removed), or some registers may be removed entirely. These changes are clearly marked within this chapter.

1.1.9.1.1 VCS_HWS_PGA — Hardware Status Page Address Register

VCS_HWS_PGA — Hardware Status Page Address Register

Register Type: MMIO
Address Offset: 04180h
Project: DevSNB+
Default Value: 0000 0000h
Access: R/W
Size (in bits): 32
Trusted Type: 1

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. [DevSNB] This address in this register is translated using the Global GTT in memory. The mapping type of the GTT entry determines the snoop nature of the transaction to memory.

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

Bit	Description
31:12	**Address** Project: DevSNB+ Security: None Address: 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** Notes: If the **Per-Process Virtual Address Space** is set, HW requires that the status page is programmed to allow for the context switch status to be reported
11:1	**Reserved** Project: DevSNB+ Format: MBZ
0	**Translation In Progress** Project: All Format: U1 FormatDesc 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.

1.2 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 GEN family. 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

The instruction format is:

MI_ARB_CHECK

Project:	All	**Length Bias:**	1
Engine:	**Video**		

The MI_ARB_CHECK instruction is used with the UHPTR register. This instruction can be used to pre-empt the current execution of the ring buffer. Note that the valid bit in the UHPTR register needs to be set for the command streamer to be pre-empted.

Programming Note:

This instruction can be placed only in a ring buffer, never 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** Project: All Format: MBZ

1.2.3 MI_ARB_ON_OFF

MI_ARB_ON_OFF			
Project:	DevSNB+	**Length Bias:**	1
Engine:	**Video**		

The MI_ARB_ON_OFF instruction is used to disable/enable context switching. 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.

This is a privileged command only if the MI_ARB_ON_OFF privileged bit is set in the VCS_MI_MODE register; it will not be effective (will be converted to a no-op) if executed from within a non-secure batch buffer. This command can only be issued when **Per-Process Virtual Address Space** is set; if the bit is set it will be converted to NOOP.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 08h MI_ARB_ON_OFF Format: OpCode
	22:1	**Reserved** Project: All 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	**Length Bias:**	1
Engine:	**Video**		
The MI_BATCH_BUFFER_END command is used to terminate the execution of commands stored in a *batch buffer* initiated using a MI_BATCH_BUFFER_START command.			

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

1.2.5 MI_CONDITIONAL_BATCH_BUFFER_END

MI_CONDITIONAL_BATCH_BUFFER_END			
Project:	DevSNB+	**Length Bias:**	2
Engine:	**Video**		
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 Note**: This command is only valid with a 1st level batch buffer (bit 22 in MI_BATCH_BUFFER_START is set to '0')			

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 36h MI_CONDITIONAL_BATCH_BUFFER_END Format: OpCode

MI_CONDITIONAL_BATCH_BUFFER_END

	22	**Use Global GTT** Project: All Default Value: 0h DefaultVaueDesc 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**. 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.
	21	**Compare Semaphore** Project: All Default Value: 0h DefaultVaueDesc Format: U1 FormatDesc If set, the value from the **Compare Data Dword** is compared to the value from the **Compare Address** in memory. If the value at **Compare Address is greater than the Compare Data Dword**, execution of current command buffer should continue. If clear, no comparison takes place.
	20	**Reserved**
	19:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2 Project: All
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** Project: All Format: MBZ

1.2.6 MI_BATCH_BUFFER_START

The MI_BATCH_BUFFER_START command format follows:

MI_BATCH_BUFFER_START		
Project: All **Default Value:** 00000000h **Engine:** **Video**		
The MI_BATCH_BUFFER_START command is used to initiate the execution of commands stored in a *batch buffer.* For restrictions on the location of batch buffers, see Batch Buffers in the Device Programming Interface chapter of *MI Functions*. The batch buffer can be specified as secure or non-secure, determining the operations considered valid when initiated from within the buffer and any attached (chained) batch buffers. See Batch Buffer Protection in the Device Programming Interface chapter of *MI Functions*.		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 31h MI_BATCH_BUFFER_START Format: OpCode
	22	**2nd Level Batch Buffer** Project: [DevSNB +] The command streamer contains 3 storage elements; 1 for the ring head address, 1 for the batch head address, and 1 for the 2nd level batch head address. When performing batch buffer chaining, hardware simply updates the head pointer of the 1st level batch address storage. There is no stack in hardware. When this bit is set, hardware uses the 2nd level batch head address storage element. Upon MI_BATCH_BUFFER_END, it will automatically return to the 1st (traditional) level batch buffer address. this allows hardware to mimic a simple 3 level stack. **Value** / **Name** / **Description** / **Project** 0h / 1st level batch / Place the batch buffer address in the 1st (traditional) level batch address storage element / [DevSNB+] 1h / 2nd level batch / Place the batch buffer address in the 2nd level batch address storage element / [DevSNB+] **Programming Notes** • A non-secure 2nd level batch buffer ***cannot*** be called from a non-secure 1st (traditional) level batch buffer. • 2nd level batch buffer chaining is ***not*** supported
	21:10	**Reserved** Project: ɪrmat: 3Z
	9	**Reserved** Project: Format:

MI_BATCH_BUFFER_START		
	8	**Buffer Security Indicator** Project: All Format: U32 When this command is executed from within a batch buffer (i.e., is a "chained" batch buffer command), this field is IGNORED and the next buffer in the chain inherits the initial buffer's security characteristics. [DevSNB+] If this bit is set, this batch buffer is non-secure and cannot execute privileged commands nor access privileged (GGTT) memory. It will be accessed via the PPGTT. If clear, this batch buffer is secure and will be accessed via the GGTT. Note that MI_STORE_DATA_IMM to non-privileged memory (via the PPGTT) *is* allowed in a non-secure batch buffer. Format = MI_BufferSecurityType 1 = MIBUFFER_NONSECURE 0 = MIBUFFER_SECURE (GGTT space) This field must be '0' unless the **Per-Process GTT Enable** is '1'
	7:0	**DWord Length** (Excludes D-Word 0,1) = 0
1	31:2	**Buffer Start Address** Format: Graphics Virtual Address[31:2] FormatDesc **Programming Notes** • A batch buffer initiated with this command must end either with a MI_BATCH_BUFFER_END command or by chaining to another batch buffer with an MI_BATCH_BUFFER_START command. • The selection of PPGTT vs. GGTT for the batch buffer is determined by the **Buffer Security Indicator** (bit 8).
	1:0	**Reserved** Project: ɪrmat: ƷZ

1.2.7 MI_FLUSH_DW

MI_FLUSH_DW			
Project:	DevSNB+	**Length Bias:**	2
Engine:	**Video**		

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).

[DevSNB]: An MI_NOOP with NOP_ID bit set must be programmed after the last MI_FLUSH_DW before head = tail

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 26h MI_FLUSH_DW Format: OpCode
	22	**Protected memory Enable** Project: DevSNB+ Format: U1 After completion of the flush, the hardware will limit all access to the Protected Content Memory. Only command streamer initiated cacheable writes are allowed to non-PCM memory.
	21	**Store Data Index** Project: DevSNB+ 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** bit is set. Else the Global HW status page is used.
	20:19	**Reserved** Project: All Format: MBZ
	18	**TLB Invalidate** Project: DevSNB+ Format: U1 If ENABLED, all TLBs will be invalidated once the flush operation is complete. Note that if the flush TLB invalidation mode is clear, a TLB invalidate will occur irrespective of this bit setting. This bit is only valid when the Post-Sync Operation field is a value of 1h or 3h.

MI_FLUSH_DW

	17	**Synchronize GFDT surface** Project: DevSNB+ 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** Project: All Format: MBZ
	15:14	**Post-Sync Operation** Project: DevSNB+ 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. — DevSNB+ 1h — Write the QWord containing Immediate Data Low, High DWs to the Destination Address — DevSNB+ 2h — **Reserved** — DevSNB+ 3h — Write the 32-bit TIMESTAMP register to the Destination Address with granularity of 640ns. Upper 32-bits are tied to '0' — DevSNB+ **Programming Notes** If executed in non-secure batch buffer, the address given will be in a PPGTT address space. If in a secure ring or batch, address given will be in GGTT space
	13:9	**Reserved** Project: All Format: MBZ
	8	**Notify Enable** Project: DevSNB+ Format: U1 If ENABLED, a Sync Completion Interrupt will be generated (if enabled by the MI Interrupt Control registers) once the sync operation is complete. See Interrupt Control Registers in *Memory Interface Registers* for details.
	7	**Video Pipeline Cache invalidate** Project: DevSNB+ Format: U1 Enable the invalidation of the video cache at the end of this flush
	6	**Reserved** Project: DevSNB+ Format:

MI_FLUSH_DW

	5:0	**DWord Length** Default Value: 2h Excludes DWord (0,1) = 2 for DWord, 3 for QWord Format: =n Total Length - 2 Project: All
1	31:3	**Address** Project: DevSNB+ Address: GraphicsAddress[31:3] Surface Type: U32 This field specifies Bits 31:3 of the Address where the DWord or QWord will be stored. Note that the address can only be QWord aligned, irrespective of data size.
	2	**Destination Address Type** Project: All Defines address space of Destination Address **Value** **Name** **Description** **Project** 0h PPGTT Use PPGTT address space for DW write All 1h GGTT Use GGTT address space for DW write All **Programming Notes** Ignored if “No write” is the selected in Operation.
	1:0	**Reserved** Project: All Format: MBZ
2..3	31:0	**Immediate Data** Format: U32 Address: GraphicsAddress[31:0] Range 0..2^32-1 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 [DevSNB A] To avoid hitting a known hardware bug, drivers cannot send a QW write when bit 5 of the address is ‘1’

1.2.8 MI_LOAD_REGISTER_IMM

The MI_LOAD_REGISTER_IMM command format is:

MI_LOAD_REGISTER_IMM			
Project:	All	**Length Bias:**	2
Engine:	**Video**		

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.

[DevSNB] The behavior of this command is controlled by Dword 3, Bit 8 (**Disable Register Access**) of the RINGBUF register. If this command is disallowed then the command stream converts it to a NOOP.

If this command is executed from a batch buffer then the behavior of this command is controlled by Dword 0, Bit 8 (**Security Indicator**) of the BATCH_BUFFER_START Command. If the batch buffer is non-secure then the command stream converts this command to a NOOP.

The following addresses should NOT be used for LRIs

1. 0x8800 - 0x88FF
2. >= 0xC0000

Limited LRI cycles to the Display Engine 0x40000-0xBFFFF) are allowed, but must be spaced to allow only one pending at a time. This can be done by issuing an SRM to the same address immediately after each LRI.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 22h MI_LOAD_REGISTER_IMM Format: OpCode
	22:12	**Reserved** Project: All Format: MBZ
	11:8	**Byte Write Disables** Project: All Format: Enable[4] (bit 8 corresponds to Data DWord [7:0]). Range: Must specify a valid register write operation. [11:8] is '1111', then the register write will not occur. [11:8] is '0000', 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 Project: All

MI_LOAD_REGISTER_IMM		
1	31:0	**Reserved** Project: All Format: MBZ
	22:2	**Register Offset** Project: All Format: U30 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** Project: All Format: MBZ
2	31:0	**Data DWord** Project: All 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	**Length Bias:**	1
Engine:	**Video**		

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 00h MI_NOOP Format: OpCode
	22:0	**Identification Number** Format: Enable FormatDesc: 1 = Write the NOP_ID register. 0 = Do not write the NOP_ID register. **Identification Number Register Write Enable:** This field enables the value in the Identification Number field to be written into the MI NOPID register. If disabled, that register is unmodified – making this command an effective “no operation” function.

MI_NOOP		
	21:0	**Identification Number** Project: All Format: U22 This field contains a 22-bit number which can be written to the MI NOPID register.

1.2.10 MI_REPORT_HEAD

The format of the MI_REPORT_HEAD command is:

MI_REPORT_HEAD			
Project:	All	**Length Bias:**	1
Engine:	**Video**		

The MI_REPORT_HEAD command causes the Head Pointer value of the ring buffer to be written to a cacheable (snooped) system memory location.

When the **Per-Process Virtual Address Space** bit is reset, the location written is relative to the address programmed in the Hardware Status Page Address Register.

Programming Notes:

- This command must not be executed from a Batch Buffer (Refer to the description of the HWS_PGA register).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 07h MI_REPORT_HEAD Format: OpCode
	22:0	**Reserved** Project: All Format: MBZ

1.2.11 MI_SEMAPHORE_MBOX

MI_SEMAPHORE_MBOX			
Project:	DevSNB+	**Length Bias:**	2
Engine:	**Video**		

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.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 16h MI_SEMAPHORE_MBOX Format: OpCode
	22	**Use Global GTT** Project: All Format: U32 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.
	21	**Update Semaphore** Project: All Format: U32 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.
	20	**Compare Semaphore** Project: All Format: U32 If set, the value from the **Semaphore Data Dword** is compared to the value from the **Semaphore Address** in memory. If the value at **Semaphore Address is greater than the Semaphore Data Dword**, execution is continued from the current command buffer. If clear, no comparison takes place. **Update Semaphore** *must* be set in this case.

MI_SEMAPHORE_MBOX		
	19	**Reserved** Project: All Format: MBZ
	18	**Compare Register** Project: DevSNB+ Format: Compare Type If set, data in MMIO register will be used for compare. If clear, data in memory will be used for compare.
	17:16	**Register Select** Project: DevSNB+ Format: Register Select: If compare register is set in bit[18], this filed indicate which register will be used. 0: BCS register (VBSYNC) 1: [Reserved] 2: CS regiser (VRSYNC) 3: Reserved
	15:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:0	**Semaphore Data Dword** Project: All Format: U32 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.
2	31:2	**PointerBitFieldName/MMIO Register Address** Project: All Address: GraphicsVirtualAddress[31:2] Surface Type: Semaphore 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.
	1:0	**Reserved** Project: All Format: MBZ

1.2.12 MI_STORE_REGISTER_MEM

MI_STORE_REGISTER_MEM

Project:	DevSNB+	**Length Bias:**	2
Engine:	**Video**		

The MI_STORE_REGISTER_MEM command requests a register read from a specified memory mapped register location in the device and store of that DWord to memory. The register address is specified along with the command to perform the read.

Programming Notes:

- The command temporarily halts command execution.
- The memory address for the write is snooped on the host bus.
- This command will cause undefined data to be written to memory if given register addresses for the PGTBL_CTL_0 or FENCE registers

The following addresses should NOT be used for SRMs

1. 0x8800 - 0x88FF
2. >= 0x40000

The only exception is an SRM cycle to 0x40000-0xBFFFF when used as part of the LRI read-after-write requirement.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 24h MI_STORE_REGISTER_MEM Format: OpCode
	22	**Use Global GTT** Project: All This bit *must* be '1' if the **Per Process GTT Enable** bit is clear. **Value** / **Name** / **Description** / **Project** 0h / Per Process Graphics Address / / All 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. / All **Programming Notes:** [DevSNB] This will be ignored and treated as if clear when executing from a PPGTT (i.e. runlist mode "non-secure") batch buffer
	21:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 1h Excludes DWord (0,1) Format: =n Total Length - 2

MI_STORE_REGISTER_MEM

1	31:23	**Reserved** Project: DevSNB+ Format: MBZ
	22:2	**Register Address** Project: All Address: MMIO Address[22:2] Surface Type: MMIO Register This field specifies Bits 22:2 of the Register offset the DWord will be read from. As the register address must be DWord-aligned, Bits 1:0 of that address MBZ. **Programming Notes** — **Project** Storing a VGA register is not permitted and will store an UNDEFINED value. — All 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. — All
	1:0	**Reserved** Project: All Format: MBZ
2	31:2	**Memory Address** Project: DevSNB+ Address: GraphicsAddress[31:2] Surface Type: 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** Project: All Format: MBZ

1.2.13 MI_STORE_DATA_IMM

The MI_STORE_DATA_IMM command format is:

MI_STORE_DATA_IMM			
Project:	All	**Length Bias:**	2
Engine:	**Video**		

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

Programming Notes:

This command 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.

[DevSNB] **Use Global GTT** will be ignored and treated as if clear when executing from a PPGTT (i.e. runlist mode "non-secure") 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** Project: All 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. **Programming Note:** [DevSNB] This bit will be ignored and treated as if clear when executing from a non-privileged batch buffer
	21:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: Excludes DWord (0,1) = 3 for QWord, 2 for DWord Format: Total Length - 2

MI_STORE_DATA_IMM		
1	31:0	**Reserved** Project: All Format: MBZ
2	31:2	**Address** Format: Bits[31:2] of a Graphics Virtual Address 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** Project: All 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).

1.2.14 MI_STORE_DATA_INDEX

The MI_STORE_DATA_INDEX command format is:

MI_STORE_DATA_INDEX			
Project:	All	**Length Bias:**	2
Engine:	**Video**		

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

Programming Notes:

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 21h MI_STORE_DATA_INDEX Format: OpCode
	22	**Reserved** Project: All Format: MBZ
	21	**Use Per-Process Hardware Status Page** Project: All Format: If this bit is set, this command will index into the per-process hardware status page at offset 20K from the LRCA. If clear, the Global Hardware Status Page will be indexed. All other devices: Reserved: MBZ. **Programming Notes:** [DevSNB] This will be ignored and treated as if set when executing from a PPGTT batch buffer
	20:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) = 2 for QWord Format: =n Total Length - 2 Project: All

MI_STORE_DATA_INDEX

1	31:12	**Reserved** Project: Format: MBZ
	11:2	**Offset** Project: All Format: U10 FormatDesc; zero-based DWord offset into the HW status page Address: GraphicsAddress[31:0] Surface Type: U32 Range [16, 1023]. This field specifies the offset (into the hardware status page) to which the data will be written. Note that the first few DWords of this status page are reserved for special-purpose data storage – targeting these reserved locations via this command is UNDEFINED. For a QWord write, the offset is valid down to bit 3 only.
	1:0	**Reserved** Project: Format: MBZ
2	31:0	**Data DWord 0** Format: U32 FormatDesc This field specifies the upper DWord value to be written to the targeted QWord location (DW 1).
3	31:0	**Data Word 1** Format: U32 FormatDesc This field specifies the upper DWord value to be written to the targeted QWord location (DW 1).

1.2.15 MI_SUSPEND_FLUSH

MI_SUSPEND_FLUSH			
Project:	All	**Length Bias:**	1
Engine:	**Video**		
Blocks MMIO sync flush or any flushes related to VT-d while enabled.			

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 0Bh MI_SUSPEND_FLUSH Format: OpCode
	22:1	**Reserved** Project: All Format: MBZ
	0	**Suspend Flush** Project: All Default Value: 0h DefaultVaueDesc Format: Enable FormatDesc This field suspends flush due to sync flush or implicit flush generated during VTD enable, disable and IOTLB invalidation. **Value** **Name** **Description** **Project** 0h Disable All 1h Enable All

1.2.16 MI_USER_INTERRUPT

MI_USER_INTERRUPT			
Project:	All	**Length Bias:**	1
Engine:	**Video**		
The MI_USER_INTERRUPT command is used to generate a User Interrupt condition. The parser will continue parsing after processing this command.			

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

1.2.17 MI_UPDATE_GTT

1.2.17.1 MI_UPDATE_GTT [DevSNB]

MI_UPDATE_GTT			
Project:	DevSNB	**Length Bias:**	2
Engine:	Video		

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, since work associated with preceding commands that are still in the pipeline may be referencing GTT entries that will be changed by its execution. The flush 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. This command will be converted to a no-op and an error flagged if it is executed from within a non-secure batch buffer.

MI_UPDATE_GTT contents must be in address/data pair. This is different from the render CS definition. PPGTT updates cannot be done via MI_UPDATE_GTT, gfx driver will have to use storeDW for PPGTT inline updates.

Note that MI_UPDATE_GTT is mainly for the pages that are strictly used by PG. If driver chooses to update the CPU used pages thru MI_UPDATE_GTT, it needs to write to MMIO address x101008 (any value) to ensure system agent TLBs are invalidated before the new pages can be used.

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** Project: All Reserved: Must be 1h. Updating Per Process Graphics Address is not supported

Value	Name	Description	Project
0h	Per Process Graphics Address	Illegal, not supported.	All
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.	All

MI_UPDATE_GTT		
	21:6	**Reserved** Project: All Format: MBZ
	5:0	**DWord Length** Default Value: 1h Excludes DWord (0,1) Format: =n Total Length – 2, max 61
1..n+1	63:44	**Entry Address** Project: All Address: 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.
	43:32	**Reserved** Project: All Format: MBZ
	31:0	**Entry Data** Project: All Format: Page Table Entry This Dword becomes the new page table entry. See PPGTT/Global GTT Table Entries (PTEs) in Memory Interface Registers.

1.2.18 MI_WAIT_FOR_EVENT

MI_WAIT_FOR_EVENT			
Project:	All	**Length Bias:**	1
Engine:	**Video**		

The MI_WAIT_FOR_EVENT command is used to pause command stream processing 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.

The effect of the wait operation depends on the source of the command. If executed from a batch buffer, the parser will halt (and suspend command arbitration) until the event/condition occurs. If executed from a ring buffer, further processing of that ring will be suspended, although command arbitration (from other rings) will continue. Note that if a specified condition does not exist (the condition code is inactive) at the time the parser executes this command, the parser proceeds, treating this command as a no-operation.

If execution of this command from a primary ring buffer causes a wait to occur, the active ring buffer will *effectively* give up the remainder of its time slice (required in order to enable arbitration from other primary ring buffers).

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

MI_WAIT_FOR_EVENT

	28:23	**MI Command Opcode** Default Value: 03h MI_WAIT_FOR_EVENT Format: OpCode
	22:20	**Reserved** Project: All Format: MBZ
	19:16	**Condition Code Wait Select** Project: All This field enables a wait for the duration that the corresponding condition code is active. These enable select one of 15 condition codes in the EXCC register, that cause the parser to wait until that condition-code in the EXCC is cleared. **Value** / **Name** / **Description** / **Project** 0h / Not enabled / Condition Code Wait Not Enabled / All 1h-5h / Enable / Condition Code select enabled; selects one of 5 codes, 0 – 4 / All 6h-15h / Reserved / / All **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** Project: All Format: MBZ

Revision History

Revision Number	Description	Revision Date
1.0	First 2011 OpenSource edition	May 2011

§§