Intel® OpenSource HD Graphics PRM

Volume 1 Part 3: Graphics Core – Memory Interface and Commands Render Engine

For the all new 2010 Intel Core Processor Family Programmer's Reference Manual (PRM)

March 2010

Revision 1.0

Doc Ref #: IHD_OS_V1Pt3_3_10

Revision History

Document Number	Revision Number	Description Re	vision Date
IHD_OS_V1Pt3_3_10	1.0	First Release.	March 2010

Contents

1. Render Engine Command Streamer

1.1 Registers in Render Engine

1.1.1 Introduct ion

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.

The registers detailed in this chapter are used across the Gen6 family of products and are extentions to previous projects. However, slight changes may be present in some registers (i.e., for features added or removed), or some registers may be removed entirely. These changes are clearly marked within this chapter.

1.1.2 Virtual Memory Control

1.1.2.1 PP_DCLV – PPGTT Directory Cacheline Valid Register

PP_DCLV – PPGTT Directory Cacheline Valid Register	
Register Type:	MMIO_CS
Address Offset:	2220h
Project:	All
Default Value:	0h
Access:	[
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 De	scription
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.**

1.1.3 Probe List Registers

Surface probing is a procedure performed at the beginning of a rendering sequence (command buffer) to verify that all required surfaces in a process' virtual address space are actually present in physical memory prior to beginning the sequence. A different process can then be switched to and run while the required surfaces are being brought into memory (by SW). The register work in concert with the probe commands (see Memory Interface Commands for Rendering) to provide this interface. "Slots" are the designated places in a processes' context image where probes (surface base addresses) are stored. The stored probes are used by SW to determine which surfaces a context requires, and are also used by HW to re-validate that surfaces are resident upon a context restore.

See MI_PROBE in Memory Interface Commands for Rendering for more details.

Note these register should only be used when Surface Fault Enable bit is set in GFX_MODE

This interface is used to signal page faults that occur during access of per-process virtual graphics memory. A fault of this nature will stall the 3D/Media pipeline behind the fault, and all new TLB requests from anywhere in the pipeline will be stalled. Faults are recorded in a fault log consisting of 32 fault slots. Page faults are non-recoverable events and will cause hardware to hang.

1.1.3.1 PP_PFIR – PPGTT Page Fault Indication Register

PP_PFIR – PPGTT Page Fault Indication Register	
Register Type:	MMIO_CS
Address Offset:	4510h
Project:	All
Default Value:	0000 0000h
Access:	R/WC
Size (in bits):	32
This register contains the flags for page faults. All bits should be cleared at once by writing FFFFFFFFh to this register once all faults have been serviced. No additional bits of this register will become set (signaling additional faults) between the time the page fault interrupt has been sent to the host and the time the host clears the **Fault In Service** bit indicating it is done servicing faults	
Bit De	**scription**
31:0	**Page Fault [31:0] Project: All Format: Array:Flag** **Fault indicator for page fault log index [31:0]. When set, this flag indicates that a page fault is outstanding. The invalid page address that was accessed can be read from fault entry [31:0]. SW should clear this bit by writing a '1' to it to indicate to HW that the fault has been serviced (the page has been mapped and should now be valid).**

1.1.3.2 PP_PFD[0:31] – PPGTT Page Fault Data Registers

PP_PFD[0:31] – PPGTT Page Fault Data Registers	
Register Type:	MMIO_CS
Address Offset:	4580h
Project:	All
Security:	None
Default Value:	0000 6820h
Access:	RO
Size (in bits):	32

The GTT Page Fault Log entries can be read from these registers.

4580h-4583h: Fault Entry 0
…
45FCh-45FFh: Fault Entry 31

Bit De	scription
31:12	**Fault Entry Page Address** Project: **All** Address: **GraphicsAddress[31:12]** **This RO field contains the faulting page address for this Fault Log entry. This field will contain a valid fault address only if the bit in the GTT Page Fault Indication Register corresponding with the address offset of this entry is set.**
11:0	**Reserved Project: All Format: MBZ**

1.1.3.3 BB_PREEMPT_ADDR—Batch Buffer Head Pointer Preemption Register

BB_PREEMPT_ADDR—Batch Buffer Head Pointer Preemption Register

Register Type:	MMIO_CS
Address Offset:	2148h
Project:	All
Default Value:	0000 0000h
Access:	RO
Size (in bits):	32

This register contains the current DWord-aligned Graphics Memory Address MI_ARB_CHECK in a batch buffer where the UHPTR register was valid. The value of the pointer below will be the address of the MI_ARB_CHECK that caused the head pointer to move.

This register is invalid if the previous preemption due to an MI_ARB_CHECK executed in the ring.

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

Bit De	scription
31:2	**Batch Buffer Head Pointer** Project: All Format: GraphicsAddress[31:2] This field specifies the DWord-aligned Graphics Memory Address MI_ARB_CHECK in a batch buffer where the UHPTR register was valid.
1:0	**Reserved** Project: All Format: MBZ

1.1.3.4 RING_BUFFER_HEAD_PREEMPT_REG

RING_BUFFER_HEAD_PREEMPT_REG	
Register Type:	MMIO_CS
Address Offset:	214Ch
Project:	All
Default Value:	0000 0000h
Access:	RO
Size (in bits):	32

This register contains the Head pointer offset in the ring when the last MI_ARB_CHECK command was executed and caused the head pointer to move due to the UHPTR register being valid. If the MI_ARB_CHECK is executed as part of the batch buffer then the value of the register will be the offset in the ring of the command past the batch buffer start that contained the MI_ARB_CHECK.

Programming Restriction:

This register should NEVER be programmed by driver, this is for HW internal use only.

Bit De	scription
31:21	**Reserved** Project: All Format: MBZ
20:2	**Preempted Head Offset** Project: All Format: U19 DWord Offset This field contains the Head pointer offset in the ring when the last MI_ARB_CHECK command was executed and caused the head pointer to move due to the UHPTR register being valid.
1	**Reserved** Project: All Format: MBZ

RING_BUFFER_HEAD_PREEMPT_REG

0 **Ring/Batch Indicator** Project: All Format: Enabled

Value Name	Description	Project
0h Ring	MI_ARB_CHECK was executed in ring and caused head pointer to be updated.	All
1h Batch	MI_ARB_CHECK was executed in batch and caused head pointer to be updated.	All

1.1.4 Mode and Misc Ctrl Registers

1.1.4.1 GFX_MODE – Graphics Mode Register

GFX_MODE	
Register Type:	MMIO
Project:	All
Default Value:	
Access:	R/W
Size (in bits):	32
Trusted Type:	1
This register contains a control bit for the 2-level PPGTT functions. This register is not saved/restored with context. This register is not reset with single-engine GFX reset; it is only reset by a global graphics reset (all engines including display).	
Bit	**Description**
31:16	**Mask Bits** Format: Mask[15:0] **Must be set to modify corresponding bit in Bits 15:0. (All implemented bits)**
14:0	**Reserved Project: All Format: MBZ**
12	**Surface Fault Enable Project: All Format: U1** **When set, surface and page fault will be handled in HW. It is undefined to use MI_PROBE and MI_UNPROBE if this bit is clear** **0: surface/page fault handling disabled (default)**

11

Replay Mode

Project: **All**

Default Value: 1h **midtriangle**

Mask: **MMIO(0x2000)#16**

Format: U1 **Context Switch Granularity**

This field controls the granularity of the replay mechanism when coming back into a previously preempted context.

Value Na	me	Description	Project
0h	mid-triangle preemption	Super span Level. Pipeline is not flushed. This implies commands parsed are executed speculatively and may not complete before a context switch.	**All**
1h mid-c	**mdbuffer preemption**	**Drawcall Level. Pipeline is flushed before switching to the next context. Commands parsed are commited to completing before a context switch**	**All**

Programming Notes

- **A fixed function pipe flush is required before modifying this field**

Unless pre-emption at a mid-triangle is required the bit must be set.

10

9

Per-Process GTT Enable

Project: **All**

Default Value: 0h **Disabled**

Format: **Enabled** **Per-Process GTT Enable**

Value	Name	Description	Project
0h	PPGTT Disable	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 in Basic Scheduler Mode.	**All**
1h	**PPGTT Enable**	**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. The PD Offset and PD Cacheline Valid registers must be set in all pipes (blitter, MFX, render) before any workload is submitted to hardware. This mode enables support for big pages (32k)**	**All**

8 **Reserved** **Project:** **All** **Format:** **MBZ**

1.1.4.2 INSTPM—Instruction Parser Mode Register

INSTPM—Instruction Parser Mode Register

Register Type:	MMIO_CS
Address Offset:	20C0h
Project:	All
Default Value:	00000000h
Access:	R/W
Size (in bits):	32
Trusted Type:	1

The INSTPM register is used to control the operation of the 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:

- If an instruction type is disabled, the parser will read those instructions but not process them.
- Error checking will be performed even if the instruction is ignored.
- All Reserved bits are implemented.
- This Register is saved and restored as part of Context.

Bit De	scription
31:16	**Mask Bits** Format: Mask[15:0] **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.
15:12	**Reserved** Project: All Format: MBZ
11	**CLFLUSH Toggle** Project: All Format: U32 BitFieldDesc
10	This bit changes polarity each time the MI_CLFLUSH command completes
9:7	**Reserved** Project: All Format: MBZ
6	**CONSTANT_BUFFER Address Offset Disable** Project: All Format: U1 When this bit is set, the 3DSTATE_CONSTANT_* Buffers' Starting Address is used as a true GraphicsAddress (not an offset). No bounds checking will be performed during access.Format = Disable

INSTPM—Instruction Parser Mode Register	
5	**Sync Flush Enable** Project: All Format: U1 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 Rings** bit in register **MI_MODE**. Only after observing **Rings Idle** set in **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 Rings**. Software is expected to follow restriction above or not use Sync flush Format = Enable (cleared by HW)
3	**Blt Instruction Disable** Project: All Format: U1 This bit instructs the Renderer instruction parser to parse and error-check BLT instructions, but not execute them. Format = Disable
2	**3D Rendering Instruction Disable** Project: All Format: U1 This bit instructs the Renderer instruction parser to parse and error-check 3D Rendering instructions, but not execute them. This bit must always be set by software if **3D State Instruction Disable** is set. Setting this bit *without* setting **3D State Instruction Disable** *is* allowed. Format = Disable
1	**3D State Instruction Disable** Project: All Format: U1 This bit instructs the Renderer instruction parser to parse and error-check 3D State instructions, but not execute them. This bit should *not* be set unless **3D Rendering Instruction Disable** (bit 2) is also set. Format = Disable
0	**Texture Palette Load Instruction Disable** Project: All Format: U1 This bit instructs the Renderer instruction parser to parse and error-check Texture Palette Load instructions, but not execute them. Format = Disable

1.1.4.3 EXCC—Execute Condition Code Register

EXCC—Execute Condition Code Register	
Register Type:	MMIO_CS
Address Offset:	2028h
Project:	All
Default Value:	00000000h
Access:	R/W,RO
Size (in bits):	32
Trusted Type:	1

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.

Bit De	scription
31:16	**Mask Bits** Format: Mask[15:0] **These bits serves 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.**
15:12	**Reserved Project: All Format: MBZ**
11	**Pending Indirect State Dirty Bit Project: All Format: U32** **This field keeps track of whether or not an indirect state pointer command has been parsed in the current context. Clears either on a context save or explicitly through a flush command**
10:7	**Pending Indirect State Counter Project: All Format: U32** **This field keeps track of the maximum number of indirect state pointers pending in the system. When the register is saved/restored, it saves either a value of 1 or 0.** **This field is Read-Only**
6:5	**Reserved Project: All 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).**

1.1.4.4 NOPID — NOP Identification Register

NOPID — NOP Identification Register					
Register Type:	MMIO_CS				
Address Offset:	2094h				
Project:	All				
Default Value:	00000000h				
Access:	RO				
Size (in bits):	32				
Trusted Type:	1				
The NOPID register contains the Noop Identification value specified by the last MI_NOOP instruction that enabled this register to be updated. [DevSNB A] This register *cannot* be used when enabling the RC6 graphics power state. As an alternative, the ring can program MI_LOAD_REGISTER_IMM to offset 0x21AC, 19:0 instead of MI_NOOP					
Bit De	**scription**				
31:22	**Reserved**	**Project:**	**All**	**Format:**	**MBZ**

1.1.4.5 FBC RT BASE ADDRESS REGISTER

FBC_RT_BASE_ADDR_REGISTER	
Register Type:	MMIO
Address Offset:	2128h [All]
Project:	All
Default Value:	--
Access:	Read/32 bit Write
Size (in bits):	32
This Register is saved and restored as part of Context.	

Bit De	scription
31:12	**4KB aligned Base4KB aligned Base Address as mapped in the PPGTT OR in the GGTT. For the render target. This base address must be the one that is either front buffer or the back-buffer (a flip target). It can be only programmed once per context. It must be programmed before any draw call binding that render target base address.Address as mapped in the PPGTT OR in the GGTT. For the render target. This register must be programmed. This base address must be the one that is either front buffer or the back-buffer (a flip target). It can be only programmed once per context. It must be programmed before any draw call binding that render target base address.** Format: **Base Address[31:12]** **Must be set to modify corresponding data bit. Reads to this field returns zero.**
11:2	**Reserved Project: All Format: MBZ**
1	**FBC Front Buffer Target** Project: **ILK** Default Value: **0h** Format: **Enable**
0	**PPGTT Render Target Base Address Valid for FBC** Project: **ILK** Default Value: **0h** Format: **Enable**

Value	Name	Description	Project
0h		FBC is targeting the Back Buffer for compression. This buffer **can** be cached in the MLC/LLC, so a GFDT flush is required before FBC can begin compression.	**ILK+**
1h		**FBC is targeting the Font Buffer for compression. This buffer cannot be cached in the MLC/LLC. FBC compression can begin after any RC flush.**	**ILK+**

Value	Name	Description	Project

FBC_RT_BASE_ADDR_REGISTER				
	0h		Base address in this register [31:12] is not valid and therefore FBC will not get any modifications from rendering.	**ILK+**
	1h		**Base address in this register [31:12] is valid and HW needs to compare the current render target base address with this base address to provide modifications to FBC.**	**ILK+**

1.1.4.6 RVSYNC – Render/Video Semaphore Sync Register

RVSYNC – Render/Video Semaphore Sync Register	
Register Type:	MMIO_CS
Address Offset:	2040h
Project:	All
Default Value:	00000000h
Access:	R/W
Size (in bits):	32
Trusted Type:	1
This register is written by VCS, read by CS.	

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

1.1.5 RINGBUF — Ring Buffer Registers

See the “Device Programming Environment” chapter for detailed information on these registers

1.1.5.1 RING_BUF FER_TAIL

RING_BUFFER_TAIL	
Register Type:	MMIO_CS
Address Offset:	2030h
Project:	All
Default Value:	00000000h
Access:	R/W
Size (in bits):	32

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

Bit De	scription
31:21	**Reserved Project: All Format: MBZ**
20:3	**Tail Offset** Project: **All** Format: U18 **QWord 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.**
2:0	**Reserved Project: All Format: MBZ**

1.1.5.2 RING_BUF FER_HEAD

RING_BUFFER_HEAD	
Register Type:	MMIO_CS
Address Offset:	2034h
Project:	All
Default Value:	00000000h
Access:	R/W
Size (in bits):	32

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

Bit De	scription
31:21	**Wrap Count** Project: **All** Default Value: 0h 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.**

RING_BUFFER_HEAD	
20:2	**Head Offset** Project: **All** Format: U19 **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** / **Project** **A RB can be enabled empty or containing some number of valid instructions.** / **All**
1	**Reserved Project: All Format: MBZ**
0	**Wait for Condition Indicator Project: All Format: Enabled** **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**

1.1.5.3 RING_BUF FER_START

RING_BUFFER_START

RING_BUFFER_START

Register Type:	MMIO_CS
Address Offset:	2038h
Project:	All
Default Value:	00000000h
Access:	R/W
Size (in bits):	32

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.

Bit De	scription
31:12	**Starting Address** Project: **All** Address: **GraphicsAddress[31:12]** Surface Type: **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. Per-process address space can only be used via a batch buffer with the appropriate Memory Space Select.**
11:0	**Reserved Project: All Format: MBZ**

1.1.5.4 RING_BUF FER_CONTROL

RING_BUFFER_CONTROL	
Register Type:	MMIO_CS
Address Offset:	203Ch
Project:	All
Default Value:	00000000h
Access:	R/W
Size (in bits):	32

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.

Bit De	scription
31:21	**Reserved Project: All Format: MBZ**
20:12	**Buffer Length** Project: **All** Format: U9 **Count of 4 KB pages** Range **0..1FF** **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]**
11	**RB Wait Project: All Format: Boolean** **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.**
9:3	**Reserved Project: All Format: MBZ**

<table>
<tr><th colspan="2">RING_BUFFER_CONTROL</th></tr>
<tr><td>2:1</td><td>
Automatic Report Head Pointer
Project: All
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.
<table>
<tr><th>Value Na</th><th>me</th><th>Description</th><th>Project</th></tr>
<tr><td>0h</td><td>MI_AUTOREPORT_OFF</td><td>Automatic reporting disabled</td><td>All</td></tr>
<tr><td>1h</td><td>MI_AUTOREPORT_64KB</td><td>Report every 16 pages (64KB)</td><td>All</td></tr>
<tr><td>2h</td><td>Reserved</td><td>Reserved</td><td>All</td></tr>
<tr><td>3h MI_</td><td>AUTOREPORT_128KB</td><td>Report every 32 pages (128KB)</td><td>All</td></tr>
</table>
<table>
<tr><th>Programming Notes</th><th>Project</th></tr>
<tr><td>When the Per-Process Virtual Address Space bit is set and automatic head reporting is desired, this field must be set to option 1 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 reset, reporting will behave just as on the prior devices (as documented above), and option 1 will report on 64KB boundary.</td><td>All</td></tr>
</table>
</td></tr>
<tr><td>0</td><td>
Ring Buffer Enable Project: All 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.
</td></tr>
</table>

1.1.5.5 UHPTR — Pending Head Pointer Register

UHPTR — Pending Head Pointer Register	
Register Type:	MMIO_CS
Address Offset:	2134h
Project:	All
Default Value:	0000 0000h
Access:	R/W
Size (in bits):	32

Bit De	scription
31:3	**Head Pointer Address** Project: **All** Default Value: **0h** Address: **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 Project: All Format: MBZ**
0	**Head Pointer Valid** Project: **All** Default Value: **0h** Format: **U1** **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 Na	me	Description	Project
0h		No valid updated head pointer register, resume execution at the current location in the ring buffer	**All**
1h		**Indicates that there is an updated head pointer programmed in this register**	**All**

1.1.6 Watchdog Timer Registers

These 2 registers together implement a watchdog timer. Writing ones to the control register enables the counter, and writing zeroes disables the counter. The 2nd register is programmed with a threshold value which, when reached, signals an interrupt then resets the counter to 0. Program the threshold value before enabling the counter or extremely frequent interrupts may result.

Note that the counter itself is not observable. It increments with the main render clock.

1.1.6.1 PR_CTR_CTL—Render Watchdog Counter Control

PR_CTR_CTL—Render Watchdog Counter Control	
Register Type:	MMIO_CS
Address Offset:	2178h
Project:	All
Default Value:	0000 0001h
Access:	R/W
Size (in bits):	32

Bit De	scription
31:0	**Counter logic op** Project: All Format: U32 This field specifies the action to be taken by the clock counter to generate interrupts. Writing 0 into this register causes a core render clock counter to be kicked off. Writing 1 into this register causes a core render clock counter to be stopped and reset to 0.

1.1.6.2 PR_CTR_THRSH— Render Watchdog Counter Threshold

PR_CTR_THRSH—Render Watchdog Counter Threshold	
Register Type:	MMIO_CS
Address Offset:	217Ch
Project:	All
Default Value:	0014 5855h
Access:	R/W
Size (in bits):	32

Bit De	scription
31:0	**Counter logic Threshold** Project: All Format: U32 This field specifies the threshold that the hardware checks against for the value of the render clock counter before generating an interrupt. The counter in hardware generates an interrupt when the threshold is reached, rolls over and starts counting again. The interrupt generated is the “Media Hang Notify” interrupt since this watchdog timer is intended primarily to remedy VLD hangs on the main pipeline.

1.1.6.3 PR_CTR—Render Watchdog Counter

PR_CTR—Render Watchdog Counter	
Register Type:	MMIO_CS
Address Offset:	2190h
Project:	All
Default Value:	0000 0000h
Access:	RO
Size (in bits):	32

Bit De	scription
31:0	**Counter Value** Project: All Format: U32 This register reflects the render watchdog counter value itself.

1.1.7 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 De	scription
31:9	**Reserved. MBZ** These bits may be assigned to interrupts on future products/steppings.
8	**Context Switch Interrupt:** Set when a context switch has just occurred.
7	**Page Fault:** This bit is set whenever there is a pending PPGTT (page or directory) fault.
6	**Timeout Counter Expired:** Set when the render pipe timeout counter (0x02190) has reached the timeout thresh-hold value (0x0217c).
5	**Reserved. MBZ** These bits may be assigned to interrupts on future products/steppings.
4	**PIPE_CONTROL 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.
3	**Render 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 Renderer Instruction Parser encounters an error while parsing an instruction.
2	**Sync Status:** This bit is set in the Hardware Status Page DW offset 0 when the Instruction Parser completes a flush with the sync enable bit active in the INSTPM register. The toggle event will happen after the render engine is flushed. The HW Status DWord write resulting from this toggle will cause the CPU’s view of graphics memory to be coherent as well (flush and invalidate the render cache). **It is the driver’s responsibility to clear this bit before the next sync flush with HWSP write enabled**
0	**Render Command Parser User Interrupt:** This status bit is set when an MI_USER_INTERRUPT instruction is executed on the Render Command Parser. Note that instruction execution is not halted and proceeds normally. A mechanism such as an MI_STORE_DATA instruction is required to associate a particular meaning to a user interrupt.

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	PIPE_CONTROL packet - Notify Enable	0
3	Master Error	Set when error occurs, cleared when error cleared
2	Sync Status	Toggled every SyncFlush Event
0	User Interrupt	0

1.1.7.1 HWSTAM — Hardware Status Mask Register

Hardware Status Mask Register	
Register Type:	MMIO_CS
Address Offset:	2098h
Project:	All
Default Value:	**FFFF FFFFh**
Access:	R/W, RO
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 Note: to write the interrupt to the HWSP, the corresponding IMR bit must also be clear (enabled).

Bit De	scription
31:0	**Hardware Status Mask Register** Project: All Default Value: FFFFFFFFh DefaultVaueDesc Format: Array of Masks refer to the Interrupt Control Register section for bit definitions, Reserved bits are RO

1.1.7.2 IMR—Interrupt Mask Register

IMR—Interrupt Mask Register	
Register Type:	MMIO_CS
Address Offset:	20A8h
Project:	All
Default Value:	FFFF FFFFh
Access:	R/W, RO
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 De	scription
31:0	**Interrupt Mask Bits** Project: All Default Value: FFFF FFFFh Format: Array of interrupt mask bits — Refer to Table 3-4 in 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. Reserved biths in teh Interrupt Control Register are RO

Value Na	me	Description	Project
0h	Not Masked	Will be reported in the IIR	All
1h	Masked	Will not be reported in the IIR	All

1.1.7.3 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'.

The following table describes the Hardware-Detected Error bits:

Hardware-Detected Error Bits

Bit De	scription
31:5	Reserved: MBZ
4	**Page Table Error**: This bit is set when a Graphics Memory Mapping Error is detected. The cause of the error is indicated (to some extent) in the PGTBL_ER register. Note: This error indications can not be cleared except by reset (i.e., it is a fatal error). 1 = Page table error
3	**Memory Privilege Violation Error.** This bit is set if a command in a non-secure batch buffer attempts an operation to the GGTT (this can only happen in commands that contain a PPGTT vs. GGTT selector). The command will be executed as if the selector bit indicated PPGTT and parsing will continue.
2	**Command Privilege Violation Error.** This bit is set if a command classified as privileged is parsed in a non-secure batch buffer. The command will be converted to a NOOP and parsing will continue.
1	Reserved: MBZ

1.1.7.3.1 EIR — Error Identity Register

EIR — Error Identity Register	
Register Type:	MMIO_CS
Address Offset:	20B0h
Project:	All
Default Value:	0000 0000h
Access:	R/W, RO
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)).

Bit De	scription
31:16	**Reserved** Project: All Format: MBZ
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 Hardware-Detected Error Bits). 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. Reserved bits are RO.

Value Na	me	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

1.1.7.3.2 EMR—Error Mask Register

EMR—Error Mask Register				
Register Type:	MMIO_CS			
Address Offset:	20B4h			
Project:	All			
Default Value:	FFFF FFFFh			
Access:	R/W, RO			
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. Reserved bits are RO.				
Bit De	**scription**			
31:16	**Reserved** Project: All Format: MBZ			
15:0	**Error Mask Bits** Project: All Default Value: FFFF FFDFh 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 Na	**me**	**Description**	**Project**
0h	Not Masked	Will be reported in the EIR	All
1h	Masked	Will not be reported in the EIR	All

1.1.7.3.3 ESR—Error Status Register

ESR—Error Status Register	
Register Type:	MMIO_CS
Address Offset:	20B8h
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 De	scription
31:16	**Reserved** Project: All Format: MBZ
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 Na	me	Description	Project
1h	Error Condition Detected	Error Condition detected	All

1.1.8 Pipelines Statistics Counter Registers

These registers keep continuous count of statistics regarding the 3D pipeline. They are saved and restored with context but should not be changed by software except to reset them to 0 at context creation time. 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.

1.1.8.1 IA_VERTICES_COUNT — Reported Vertices Counter

IA_VERTICES_COUNT	
Register Type:	MMIO_CS
Address Offset:	2310h
Project:	All
Default Value:	00000000h; 00000000h;
Access:	R/W
Size (in bits):	64
Trusted Type:	1
This register stores the count of vertices processed by VF. This register is part of the context save and restore.	
Bit De	**scription**
63:0	**IA Vertices Count Report** Total number of vertices fetched by the VF stage. This count is updated for every input vertex as long as **Statistics Enable** is set in VF_STATE (see the Vertex Fetch Chapter in the *3D* Volume.)

1.1.8.2 IA_PRIMITIVES_COUNT — Reported Vertex Fetch Output Primitives Counter

IA_PRIMITIVES_COUNT	
Register Type:	MMIO_CS
Address Offset:	2318h
Project:	All
Default Value:	00000000h; 00000000h;
Access:	R/W
Size (in bits):	64
Trusted Type:	1
This register stores the count of primitives generated by VF. This register is part of the context save and restore.	
Bit De	**scription**
63:0	**IA Primitives Count Report** Total number of primitives output by the Vertex Fetch (IA) stage. This count is updated for every primitive *output* by the VF stage, as long as **Statistics Enable** is set in VF_STATE (see the Vertex Fetch Chapter in the *3D* Volume.)

1.1.8.3 GS_INVOCATION_COUNT — Reported Geometry Shader Thread Invocation Counter

GS_INVOCATION_COUNT	
Register Type:	MMIO_CS
Address Offset:	2328h
Project:	All
Default Value:	00000000h; 00000000h;
Access:	R/W
Size (in bits):	64
Trusted Type:	1
This register stores the number of invoked geometry shader threads. This register is part of the context save and restore.	
Bit De	**scription**
63:0	**GS Invocation Count** Number of geometry shader threads invoked by the GS stage. Updated only when **Statistics Enable** is set in GS_STATE (see the Geometry Shader Chapter in the *3D* Volume.)

1.1.8.4 GS_PRIMITIVES_COUNT — Reported Geometry Shader Output Primitives Counter

GS_PRIMITIVES_COUNT	
Register Type:	MMIO_CS
Address Offset:	2330h
Project:	All
Default Value:	00000000h; 00000000h;
Access:	R/W
Size (in bits):	64
Trusted Type:	1
This register reflects the total number of primitives that have been output by the Geometry Shader stage. This register is part of the context save and restore.	

Bit De	scription
63:0	**GS Primitives Count** Total number of primitives output by the geometry stage. Updated only when **Statistics Enable** is set in GS_STATE (see the Geometry Shader Chapter in the *3D* Volume.)

1.1.8.5 CL_INVOCATION_COUNT— Reported Clipper Thread Invocation Counter

CL_INVOCATION_COUNT	
Register Type:	MMIO_CS
Address Offset:	2338h
Project:	All
Default Value:	00000000h; 00000000h;
Access:	R/W
Size (in bits):	64
Trusted Type:	1
This register stores the count of objects entering the Clipper stage. This register is part of the context save and restore.	

Bit De	scription
63:0	**CL Invocation Count Report** Number of objects entering the clipper stage. Updated only when **Statistics Enable** is set in CLIP_STATE (see the Clipper Chapter in the *3D* Volume.)

1.1.8.6 CL_PRIMITIVES_ COUNT— Reported Clipper Output Primitives Counter

CL_PRIMITIVES_COUNT	
Register Type:	MMIO_CS
Address Offset:	2340h
Project:	All
Default Value:	00000000h; 00000000h;
Access:	R/W
Size (in bits):	64
Trusted Type:	1
This register reflects the total number of primitives that have been output by the clipper. This register is part of the context save and restore.	
Bit De	**scription**
63:0	**Clipped Primitives Output Count** Total number of primitives output by the clipper stage. This count is updated for every primitive *output* by the clipper stage, as long as **Statistics Enable** is set in SF_STATE (see the Clipper and SF Chapters in the *3D* Volume.)

1.1.8.7 PS_DEPTH_COUNT — Reported Pixels Passing Depth Test counter

PS_DEPTH_COUNT	
Register Type:	MMIO_CS
Address Offset:	2350h
Project:	All
Default Value:	00000000h; 00000000h;
Access:	R/W
Size (in bits):	64
Trusted Type:	1

This register stores the value of the count of pixels that have passed the depth test. This register is part of the context save and restore. Note that the value of this register can be obtained in a pipeline-synchronous fashion without a pipeline flush by using the 3DCONTROL command. See 3D Overview in the 3D volume.

Bit De	scription
63:0	**Depth Count** This register reflects the total number of pixels that have passed the depth test (i.e., will be visible). All pixels are counted when **Statistics Enable** is set in the Windower State. See the Windower chapter of the *3D* volume for details. Pixels that pass the depth test but fail the stencil test will *not* be counted.

1.1.8.8 TIMESTAMP — Reported Timestamp Count

TIMESTAMP — Reported Timestamp Count	
Register Type:	MMIO_CS
Address Offset:	2358h
Project:	All
Default Value:	0000 0000 0000 0000h
Access:	RO. This register is *not* set by the context restore.
Size (in bits):	64

This register provides an elapsed real-time value that can be used as a timestamp for GPU events over short periods of time. Note that the value of this register can be obtained in a 3D pipeline-synchronous fashion without a pipeline flush by using the PIPE_CONTROL command. See *3D Geometry Pipeline* in the "3D and Media" volume.

This register (effectively) counts at a constant frequency by adjusting the increment amount according to the actual reference clock frequency. SW therefore does not need to know the reference clock frequency.

This register is *not* reset by a graphics reset. It will maintain its value unless a full chipset reset is performed.

Bit De	scription
63:36	**Reserved** Project: All Format: MBZ
31:0	**BitFieldName** Project: All Format: U32 This register represents 80 ns of time.

1.1.9 SO_NUM_PRIMS_WRITTEN— Reported Stream Output Num Primitives Written Counter ([DevCTG] Only)

SO_NUM_PRIMS_WRITTEN— Reported Stream Output Num Primitives Written Counter	
Register Type:	MMIO
Address Offset:	2288h
Project:	DevCTG+
Default Value:	0000 0000 0000 0000h
Access:	R only. This register is set by the context restore.
Size (in bits):	64
This register is used to (indirectly) count the number of primitives which GS threads have successfully written to Streamed Vertex Output buffers. This register is part of the context save and restore. **[Errata]** This regiser gets reset when write happens to register 2380h	
Bit De	**scription**
63:0	**Num Prims Written Count** Project: DevCTG Format: U64 This count is incremented (by one) every time a GS thread outputs a DataPort Streamed Vertex Buffer Write message with the **Increment Num Prims Written** bit set in the message header (see the *Geometry Shader* and *Data Port* chapters in the *3D* Volume.)

1.1.10 SO_ PRIM_STORAGE_NEEDED — Reported Stream Output Primitive Storage Needed Counter ([DevCTG] Only)

SO_PRIM_STORAGE_NEEDED — Reported Stream Output Primitive Storage Needed Counter	
Register Type:	MMIO
Address Offset:	2280h
Project:	DevCTG+
Default Value:	0000 0000 0000 0000h
Access:	R only. This register is set by the context restore.
Size (in bits):	64
This register is used to (indirectly) count the number of primitives which GS threads would have written to Streamed Vertex Output buffers if all buffers had been large enough to accommodate the writes . This register is part of the context save and restore. **[Errata]** This register gets reset when write happens to register 2388h	
Bit De	**scription**
63:0	**Prim Storage Needed Count** Project: DevCTG Format: U64 This count is incremented (by one) every time a GS thread outputs a DataPort Streamed Vertex Buffer Write message with the **Increment Prim Storage Needed** bit set in the message header (see the *Geometry Shader* and *Data Port* chapters in the *3D* Volume.)

1.1.11 Display Related Registers for Flip Queue

1.1.11.1 MAXQ_FLIP_A – Maximum Flips Allowed for Display A Register ([DevCTG] Only)

MAXQ_FLIP_A – Maximum Flips Allowed for Display A Register	
Register Type:	MMIO
Address Offset:	2530h
Project:	DevCTG
Default Value:	0000 0004h
Access:	R/W
Size (in bits):	32
This register is *not* saved or restored with context. The render-only reset will not affect this register.	
Bit De	**scription**
31:5	**Reserved** Project: DevCTG Format: MBZ
4:0	**Maximum Flips allowed** Project: DevCTG Format: U5 The number of flip buffers allocated by the software to support queued flips for Display plane A. For a double buffered case, the value programmed into this register should be 1. [DevCTG]: Maximum value in this register is 8

1.1.11.2 MAXQ_FLIP_B – Maximum Flips Allowed for Display B Register ([DevCTG] Only)

MAXQ_FLIP_B – Maximum Flips Allowed for Display B Register	
Register Type:	MMIO
Address Offset:	2534h
Project:	DevCTG
Default Value:	0000 0004h
Access:	R/W
Size (in bits):	32
This register is *not* saved or restored with context. The render-only reset will not affect this register.	
Bit De	**scription**
31:5	**Reserved** Project: DevCTG Format: MBZ
4:0	**Maximum Flips allowed** Project: DevCTG Format: U5 The number of flip buffers allocated by the software to support queued flips for Display plane B. For a double buffered case, the value programmed into this register should be 1. [DevCTG]: Maximum value in this register is 8

1.1.11.3 NUM_FLIP_A – Number of flips pending on Display A Register ([DevCTG] Only)

NUM_FLIP_A – Number of flips pending on Display A Register	
Register Type:	MMIO
Address Offset:	2538h
Project:	DevCTG
Default Value:	0000 0000h
Access:	R/W [Debug only]
Size (in bits):	32
This register is *not* saved or restored with context. The render-only reset will not affect this register.	
Bit De	**scription**
31:5	**Reserved** Project: DevCTG Format: MBZ
4:0	**Number of Flips pending** Project: DevCTG Format: U5 The number of flips pending in the hardware for Display plane A.

1.1.11.4 NUM_FLIP_B – Number of flips pending on Display B Register ([DevCTG] Only)

NUM_FLIP_B – Number of flips pending on Display B Register	
Register Type:	MMIO
Address Offset:	253Ch
Project:	DevCTG
Default Value:	0000 0000h
Access:	R/W [Debug only]
Size (in bits):	32
This register is *not* saved or restored with context. The render-only reset will not affect this register.	
Bit De	**scription**
31:5	**Reserved** Project: DevCTG Format: MBZ
4:0	**Number of Flips pending** Project: DevCTG Format: U5 The number of flips pending in the hardware for Display plane B.

1.1.12 Predicate Render Registers

1.1.12.1 MI_PREDICATE_SRC0 - Predicate Rendering Temporary Register0

MI_PREDICATE_SRC0 – Predicate Rendering Temporary Register0	
Register Type:	MMIO_CS
Address Offset:	2400-2407h
Project:	All
Default Value:	0000 0000 0000 0000h
Access:	R/W
Size (in bits):	64

Bit De	scription
63:0	**MI_PREDICATE_SRC0** Project: All Format: This register is a temporary register for Predicate Rendering. See *Predicate Rendering* section for more details.

1.1.12.2 MI_PREDI CATE_SRC1– Predicate Rendering Temporary Register1

MI_PREDICATE_SRC1 – Predicate Rendering Temporary Register1	
Register Type:	MMIO_CS
Address Offset:	2408-240Fh
Project:	All
Default Value:	0000 0000 0000 0000h
Access:	R/W
Size (in bits):	64

Bit De	scription
63:0	**MI_PREDICATE_SRC1** Project: All Format: This register is a temporary register for Predicate Rendering. See *Predicate Rendering* section for more details.

1.1.12.3 MI_PREDI CATE_DATA– Predicate Rendering Data Storage

MI_PREDICATE_DATA – Predicate Rendering Data Storage	
Register Type:	MMIO_CS
Address Offset:	2410-2417h
Project:	All
Default Value:	0000 0000 0000 0000h
Access:	R/W
Size (in bits):	64

Bit De	scription
63:0	**MI_PREDICATE_DATA** Project: All Format: This register is used either as computed value based off the MI_PREDICATE_SRC0 and MI_PREDICATE_SRC1 or a temporary register. See *Predicate Rendering* section for more details.

1.1.12.4 MI_PREDICATE_RESULT – Predicate Rendering Data Result

MI_PREDICATE_RESULT – Predicate Rendering Data Result	
Register Type:	MMIO_CS
Address Offset:	2418h
Project:	All
Default Value:	0000 0000h
Access:	RO
Size (in bits):	32

Bit De	scription
31:1	**Reserved** Project: All Format: MBZ
0	**MI_PREDICATE_RESULT** Project: All Format: This bit is the result of the last MI_PREDICATE.

1.1.13 AUTO_DRAW Registers

1.1.13.1 3DPRIM_END_OFFSET – Auto Draw End Offset

3DPRIM_END_OFFSET - Auto Draw End Offset	
Register Type:	MMIO_CS
Address Offset:	2420-2423h
Project:	All
Default Value:	0000 0000h
Access:	R/W
Size (in bits):	32

Bit De	scription
31:0	**End Offset** Project: All Format: U32 This register is used to store the end offset value used by the Vertex Fetch to determine when to stop processing the 3D_PRIMITIVE command. This register is valid when the End Offset Enable is set in the 3D_PRIMITIVE command.

1.1.13.2 3DPRIM_START_VERTEX – Load Indirect Start Vertex

3DPRIM_START_VERTEX - Load Indirect Start Vertex	
Register Type:	MMIO_CS
Address Offset:	2430-2433h
Project:	All
Default Value:	0000 0000h
Access:	R/W
Size (in bits):	32

Bit De	scription
31:0	**Start Vertex** Project: All Format: U32 This register is used to store the Start Vertex of the 3D_PRIMITIVE command when Load Indirect Enable is set.

1.1.13.3 3DPRIM_VERTEX_COUNT – Load Indirect Vertex Count

3DPRIM_VERTEX_COUNT - Load Indirect Vertex Count	
Register Type:	MMIO_CS
Address Offset:	2434-2437h
Project:	All
Default Value:	0000 0000h
Access:	R/W
Size (in bits):	32

Bit De	scription
31:0	**Vertex Count** Project: All Format: U32 This register is used to store the Vertex Count of the 3D_PRIMITIVE command when Load Indirect Enable is set.

1.1.13.4 3DPRIM_INSTA NCE_COUNT – Load Indirect Instance Count

3DPRIM_INSTANCE_COUNT - Load Indirect Instance Count	
Register Type:	MMIO_CS
Address Offset:	2438-243Bh
Project:	All
Default Value:	0000 0000h
Access:	R/W
Size (in bits):	32

Bit De	scription
31:0	**Instance Count** Project: All Format: This register is used to store the Instance Count of the 3D_PRIMITIVE command when Load Indirect Enable is set.

1.1.13.5 3DPRIM_START _INSTANCE – Load Indirect Start Instance

3DPRIM_START_INSTANCE - Load Indirect Start Instance	
Register Type:	MMIO_CS
Address Offset:	243C-243Fh
Project:	All
Default Value:	0000 0000h
Access:	R/W
Size (in bits):	32

Bit De	scription
31:0	**Start Vertex** Project: All Format: U32 This register is used to store the Start Instance of the 3D_PRIMITIVE command when Load Indirect Enable is set.

1.1.13.6 3DPRIM_BA SE_VERTEX – Load Indirect Base Vertex

3DPRIM_BASE_VERTEX - Load Indirect Base Vertex	
Register Type:	MMIO_CS
Address Offset:	2440-2443h
Project:	All
Default Value:	0000 0000h
Access:	R/W
Size (in bits):	32

Bit De	scription
31:0	**Base Vertex** Project: All Format: S31 This register is used to store the Base Vertex of the 3D_PRIMITIVE command when Load Indirect Enable is set.

1.1.14 Performance Statistics Registers

1.1.14.1 OACONTROL – Observation Architecture Control

OACONTROL – Observation Architecture Control	
Register Type:	MMIO
Address Offset:	2360h
Project:	All
Default Value:	00000000h
Access:	R/W
Size (in bits):	32

This register is used to program the OA unit.

[DevSNB B {W/A}] If software intends to reset the OA buffer to start a new one, after clearing the **Timer Enable** bit, software must check to see if the head pointer in **OASTATUS2** is *greater than* the tail pointer in **OASTATUS1**. If so software must program the head pointer to a value less than the current head pointer value. This must be done *before* the buffer becomes active again

Bit De	scription
31:12	**Select Context ID** Project: All Specifies the context ID of the one context that affects the performance counters. All other contexts are ignored.
11:6	**Timer Period** Project: All Format: Select Specifies the period of the timer strobe as a function of the minimum TIME_STAMP resolution. The period is determined by selecting a specified bit from the TIME_STAMP register as follows: $StrobePeriod = MinimumTimeStampPeriod * 2^{TimerPeriod}$ The exponent is defined by this field. Note: The TIME_STAMP is not reset at start time so the phase of the strobe is not synchronized with the enable of the OA unit. This could result in approximately a full StrobePeriod elapsing prior to the first trigger. Usage for this mechanism should be time based periodic triggering, typically.

OACONTROL – Observation Architecture Control

5 **Timer Enable**

Project: All

Default Value: 0h Disabled

Format: Enable

This field enables the timer logic to output a periodic strobe, as defined by the Timer Period. When disabled the timer output is not asserted.

Value Na	me	Description	Project
0h	Disable	Counter does not get written out on regular interval	All
1h	Enable	Counter gets written out on regular intervals, defined by the **Timer Period**	All

4:2 **Counter Select**

Project: All

Default Value: 0h Write 64 bytes

Format: Counter size Select

This field when reset (i.e. bit = 0) selects the first 64B with time-stamp, REPORT_ID and 13 counters. When this bit is 1, second 64B write with 16 counters are written out.

Value Size		Description	Project
001b	128bytes	Write 128 Bytes containing: • RPT_ID, TIME_STAMP, and the A-Cntr 0-12 counters • A-Cntr 13-28 counters.	All
011b	196bytes	Write 196 Bytes containing. • RPT_ID, TIME_STAMP, and the A-Cntr 0-12 counters • A-Cntr 13-28 counters. • B-Cntr 0-3 counters. • C-Cntr 0-11 counters.	All

1 **Specific Context Enable**

Project: All

Default Value: 0h All contexts considered

Mask: MMIO(0x2000)#16

Format: U32 FormatDesc

Enables counters to work on a context specific workload. The context is given by bits 31:12

[DevSNB A] Must be set to '1' (context aware)

OACONTROL – Observation Architecture Control		
	Value Name	**Description**	**Project**
	0h	Disable	All contexts are considered	All
	1h	Enable	Only the contexts with the **Select Context ID** are considered	All
0	**Performance Counter Enable** Project: All Format: Enable Global performance counter enable. If clear, no counting will occur. MI_REPORT_PERF_COUNT is undefined when clear.		

When either the MI_REPORT_PERF_COUNT command is received or the internal Report Triggering logic fires following 64 byte cache lines are written to memory. There are five formats as defined by the Counter Select within the OACONTROL word. The RPT_ID always stored in the lowest addressed DWord.

Counter Select = 000

A-Cntr 0	A-Cntr 1	A-Cntr 2	A-Cntr 3	A-Cntr 4	TIME_STAMP		RPT_ID
A-Cntr 5	A-Cntr 6	A-Cntr 7	A-Cntr 8	A-Cntr 9	A-Cntr 10	A-Cntr 11	A-Cntr 12

Counter Select = 001

A-Cntr 0	A-Cntr 1	A-Cntr 2	A-Cntr 3	A-Cntr 4	TIME_STAMP		RPT_ID
A-Cntr 5	A-Cntr 6	A-Cntr 7	A-Cntr 8	A-Cntr 9	A-Cntr 10	A-Cntr 11	A-Cntr 12
A-Cntr 13	A-Cntr 14	A-Cntr 15	A-Cntr 16	A-Cntr 17	A-Cntr 18	A-Cntr 19	A-Cntr 20
A-Cntr 21	A-Cntr 22	A-Cntr 23	A-Cntr 24	A-Cntr 25	A-Cntr 26	A-Cntr 27	A-Cntr 28

Counter Select = 010

A-Cntr 0	A-Cntr 1	A-Cntr 2	A-Cntr 3	A-Cntr 4	TIME_STAMP		RPT_ID
A-Cntr 5	A-Cntr 6	A-Cntr 7	A-Cntr 8	A-Cntr 9	A-Cntr 10	A-Cntr 11	A-Cntr 12
C-Cntr 3	C-Cntr 2	C-Cntr 1	C-Cntr 0	B-Cntr 3	B-Cntr 2	B-Cntr 1	B-Cntr 0
C-Cntr 11	C-Cntr 10	C-Cntr 9	C-Cntr 8	C-Cntr 7	C-Cntr 6	C-Cntr 5	C-Cntr 4

Counter Select = 011

A-Cntr 0	A-Cntr 1	A-Cntr 2	A-Cntr 3	A-Cntr 4	TIME_STAMP		RPT_ID
A-Cntr 5	A-Cntr 6	A-Cntr 7	A-Cntr 8	A-Cntr 9	A-Cntr 10	A-Cntr 11	A-Cntr 12
A-Cntr 13	A-Cntr 14	A-Cntr 15	A-Cntr 16	A-Cntr 17	A-Cntr 18	A-Cntr 19	A-Cntr 20
A-Cntr 21	A-Cntr 22	A-Cntr 23	A-Cntr 24	A-Cntr 25	A-Cntr 26	A-Cntr 27	A-Cntr 28
C-Cntr 3	C-Cntr 2	C-Cntr 1	C-Cntr 0	B-Cntr 3	B-Cntr 2	B-Cntr 1	B-Cntr 0
C-Cntr 11	C-Cntr 10	C-Cntr 9	C-Cntr 8	C-Cntr 7	C-Cntr 6	C-Cntr 5	C-Cntr 4

Counter Select = 100

C-Cntr 3	C-Cntr 2	C-Cntr 1	C-Cntr 0	INST ADD	TIME_STAMP		RPT_ID
C-Cntr 11	C-Cntr 10	C-Cntr 9	C-Cntr 8	C-Cntr 7	C-Cntr 6	C-Cntr 5	C-Cntr 4

1.1.14.2 OASTATUS1 – Observation Architecture Status Register

OASTATUS1—Observation Architecture Status Register	
Register Type:	MMIO
Address Offset:	2364h
Project:	All
Default Value:	00000000h
Access:	R/W
Size (in bits):	32

This register is used to program the OA unit.

Bit De	scription
31:6	**Tail Ppointer** Project: All Virtual address of the internal trigger based buffer and it is updated for every 64B cacheline write to memory when reporting via internal trigger. This pointer will not be updated for MI_REPORT_PERF_COUNT command based writes. When OA is enabled, this address must be programmed by SW to the base address of the internal trigger base mechanism.
5:3	**Inter Trigger Report Buffer Size** Project: All Default Value: 0h All context considered This field indicates the size of buffer for internal trigger mechanism. This field is programmed in terms of multiple of 4 pages (i.e. 16KB).

Value De	scription	Project
0b	16KB	All
1b	32KB	All
2	48KB	All
3	64KB	All
4	80KB	All
5	96KB	All
6	112KB	All
7	128KB	All

OASTATUS1—Observation Architecture Status Register	
2	**Counter OverFlow Error** Project: All Format: Select This bit is set if any of the counters overflows. This bit can be reset by SW in B0.
1	**Buffer Overflow** Project: All Default Value: 0h This bit is set when the Tail-pointer - Head pointer > max internal trigger buffer size
0	**Report Lost Error** Project: All Format: Enable This bit is set if the Report Logic is requested to write out the counter values before the previous report request was completed. The report request is ignored and the counter continue to count. This bit can be reset by SW in B0.

1.1.14.3 OASTATUS2 – Observation Architecture Status Register

OASTATUS2—Observation Architecture Status Register	
Register Type:	MMIO
Address Offset:	2368h
Project:	All
Default Value:	00000000h
Access:	RO
Size (in bits):	32
This register is used to program the OA unit.	
Bit De	**scription**
31:6	**Head Pointer** Project: All Virtual address of the internal trigger based buffer that is updated by software after consuming from the report buffer. This pointer must be updated by SW for internal trigger base buffer only.
5:0	**Reserved** Project: All Format: MBZ

1.1.14.4 OABUFFER – Observation Architecture Buffer

OABUFFER— Observation Architecture Status Register						
Register Type:	MMIO					
Address Offset:	23B0h					
Project:	All					
Default Value:	00000000h					
Access:	RW					
Size (in bits):	32					
This register is used to program the OA unit. [DevSNB A{W/A}] This offset does not exist. Instead, the value is set during the tail address MMIO write to the same data value as the tail address (0x2364). [DevSNB C+] This MMIO must be set *before* the **OASTATUS1** and **OASTATUS2** registers						
Bit De	**scription**					
31:6	**Report Buffer Offset** Project: All This field specifies 64B aligned GFX MEM address where the chap counter values are reported.					
5:0	**Reserved**	Project:	All	Format:	MBZ	

1.1.14.5 OASTARTTRIG1 – Observation Architecture Start Trigger

OASTARTTRIG1—Observation Architecture Buffer	
Register Type:	MMIO
Address Offset:	238Ch
Project:	All
Default Value:	00000000h
Access:	RW
Size (in bits):	32
This register is used to program the OA unit.	
Bit De	**scription**
31:16	**Reserved** Project: All Format: MBZ
15:0	**Threshold Value** Project: All Format: U16 Threshold value for the compare logic within the trigger logic

1.1.14.6 OASTARTTRIG2 – Observation Architecture Start Trigger

OASTARTTRIG2—Observation Architecture Start Trigger	
Register Type:	MMIO
Address Offset:	2388h
Project:	All
Default Value:	00000000h
Access:	RW
Size (in bits):	32
This register is used to program the OA unit.	
Bit De	**scription**
31	**event select 3, to select between Boolean and NOA event for the counter 4 to count** **0 NOA** **1 Boolean**
30	**event select 2, to select between Boolean and NOA event for the counter 3 to count** **0 NOA** **1 Boolean**
29	**event select 1, to select between Boolean and NOA event for the counter 2 to count** **0 NOA** **1 Boolean**

OASTARTTRIG2—Observation Architecture Start Trigger	
28	**event select 0, to select between Boolean and NOA event for the counter 1 to count** **0 NOA** **1 Boolean**
27:24	**Reserved**
23	**Threshold Enable** Enable the threshold compare logic within the trigger logic.
22	**Invert D Enable 0** Invert the specified signal at the D stage of the trigger logic.
21	**Invert C Enable 1** Invert the specified signal at the C stage of the trigger logic.
20	**Invert C Enable 0** Invert the specified signal at the C stage of the trigger logic.
19	**Invert B Enable 3** Invert the specified signal at the B stage of the trigger logic.
18	**Invert B Enable 2** Invert the specified signal at the B stage of the trigger logic.
17	**Invert B Enable 1** Invert the specified signal at the B stage of the trigger logic.
16	**Invert B Enable 0** Invert the specified signal at the B stage of the trigger logic.
15	**Invert A Enable 15** Invert the specified signal at the A stage of the trigger logic.
14	**Invert A Enable 14** Invert the specified signal at the A stage of the trigger logic.
13	**Invert A Enable 13** Invert the specified signal at the A stage of the trigger logic.

OASTARTTRIG2—Observation Architecture Start Trigger	
12	**Invert A Enable 12** Invert the specified signal at the A stage of the trigger logic.
11	**Invert A Enable 11** Invert the specified signal at the A stage of the trigger logic.
10	**Invert A Enable 10** Invert the specified signal at the A stage of the trigger logic.
9	**Invert A Enable 9** Invert the specified signal at the A stage of the trigger logic.
8	**Invert A Enable 8** Invert the specified signal at the A stage of the trigger logic.
7	**Invert A Enable 7** Invert the specified signal at the A stage of the trigger logic.
6	**Invert A Enable 6** Invert the specified signal at the A stage of the trigger logic.
5	**Invert A Enable 5** Invert the specified signal at the A stage of the trigger logic.
4	**Invert A Enable 4** Invert the specified signal at the A stage of the trigger logic.
3	**Invert A Enable 3** Invert the specified signal at the A stage of the trigger logic.
2	**Invert A Enable 2** Invert the specified signal at the A stage of the trigger logic.
1	**Invert A Enable 1** Invert the specified signal at the A stage of the trigger logic.
0	**Invert A Enable 0** Invert the specified signal at the A stage of the trigger logic.

1.1.14.7 OASTARTTRIG3 – Observation Architecture Start Trigger

OASTARTTRIG3— Observation Architecture Start Trigger					
Register Type:	MMIO				
Address Offset:	2384h				
Project:	All				
Default Value:	00000000h				
Access:	RW				
Size (in bits):	32				
This register is used to program the OA unit.					
Bit De	**scription**				
31:28	**NOA Signal Select 15** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
27:24	**NOA Signal Select 14** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
23:20	**NOA Signal Select 13** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
19:16	**NOA Signal Select 12** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
15:12	**NOA Signal Select 11** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
11:8	**NOA Signal Select 10** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
7:4	**NOA Signal Select 9** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
3:0	**NOA Signal Select 8** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4

1.1.14.8 OASTARTTRIG4 – Observation Architecture Start Trigger

OASTARTTRIG4—Observation Architecture Start Trigger					
Register Type:	MMIO				
Address Offset:	2380h				
Project:	All				
Default Value:	00000000h				
Access:	RW				
Size (in bits):	32				
This register is used to program the OA unit.					
Bit De	**scription**				
31:28	**NOA Signal Select 7** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
27:24	**NOA Signal Select 6** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
23:20	**NOA Signal Select 5** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
19:16	**NOA Signal Select 4** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
15:12	**NOA Signal Select 3** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
11:8	**NOA Signal Select 2** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
7:4	**NOA Signal Select 1** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
3:0	**NOA Signal Select 0** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4

1.1.14.9 OAREPORTTRIG1 – Observation Architecture Report Trigger

OAREPORTTRIG1— Observation Architecture Report Trigger

Register Type: MMIO
Address Offset: 237Ch
Project: All
Default Value: 00000000h
Access: RW
Size (in bits): 32

This register is used to program the OA unit.

Bit De	scription
31:16	**Occurrence vs. Duration Select** Project: All Format: Occurrence[16] 1 bit per NOA counter total 16 bits
15:0	**Threshold Value** Project: All Format: U16 Threshold value for the compare logic within the trigger logic

Value Na	me	Description	Project
0h	Duration		All
1h	Occurence		All

1.1.14.10 OAREPORTTRIG2 – Observation Architecture Report Trigger

OAREPORTTRIG2— Observation Architecture Report Trigger	
Register Type:	MMIO
Address Offset:	2378h
Project:	All
Default Value:	00000000h
Access:	RW
Size (in bits):	32
This register is used to program the OA unit.	

Bit De	scription
31:24	**Reserved** Project: All Format: MBZ
23	**Threshold Enable** Enable the threshold compare logic within the trigger logic.
22	**Invert D Enable 0** Invert the specified signal at the D stage of the trigger logic.
21	**Invert C Enable 1** Invert the specified signal at the C stage of the trigger logic.
20	**Invert C Enable 0** Invert the specified signal at the C stage of the trigger logic.
19	**Invert B Enable 3** Invert the specified signal at the B stage of the trigger logic.
18	**Invert B Enable 2** Invert the specified signal at the B stage of the trigger logic.
17	**Invert B Enable 1** Invert the specified signal at the B stage of the trigger logic.
16	**Invert B Enable 0** Invert the specified signal at the B stage of the trigger logic.
15	**Invert A Enable 15** Invert the specified signal at the A stage of the trigger logic.

OAREPORTTRIG2— Observation Architecture Report Trigger	
14	**Invert A Enable 14** Invert the specified signal at the A stage of the trigger logic.
13	**Invert A Enable 13** Invert the specified signal at the A stage of the trigger logic.
12	**Invert A Enable 12** Invert the specified signal at the A stage of the trigger logic.
11	**Invert A Enable 11** Invert the specified signal at the A stage of the trigger logic.
10	**Invert A Enable 10** Invert the specified signal at the A stage of the trigger logic.
9	**Invert A Enable 9** Invert the specified signal at the A stage of the trigger logic.
8	**Invert A Enable 8** Invert the specified signal at the A stage of the trigger logic.
7	**Invert A Enable 7** Invert the specified signal at the A stage of the trigger logic.
6	**Invert A Enable 6** Invert the specified signal at the A stage of the trigger logic.
5	**Invert A Enable 5** Invert the specified signal at the A stage of the trigger logic.
4	**Invert A Enable 4** Invert the specified signal at the A stage of the trigger logic.
3	**Invert A Enable 3** Invert the specified signal at the A stage of the trigger logic.
2	**Invert A Enable 2** Invert the specified signal at the A stage of the trigger logic.

OAREPORTTRIG2— Observation Architecture Report Trigger	
1	**Invert A Enable 1** Invert the specified signal at the A stage of the trigger logic.
0	**Invert A Enable 0** Invert the specified signal at the A stage of the trigger logic.

1.1.14.11 OAREPORTTRIG3 – Observation Architecture Report Trigger

OAREPORTRIG3— Observation Architecture Report Trigger					
Register Type:	MMIO				
Address Offset:	2374h				
Project:	All				
Default Value:	00000000h				
Access:	RW				
Size (in bits):	32				
This register is used to program the OA unit.					
Bit De	**scription**				
31:28	**NOA Signal Select 15** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
27:24	**NOA Signal Select 14** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
23:20	**NOA Signal Select 13** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
19:16	**NOA Signal Select 12** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
15:12	**NOA Signal Select 11** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
11:8	**NOA Signal Select 10** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
7:4	**NOA Signal Select 9** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
3:0	**NOA Signal Select 8** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4

1.1.14.12 OAREPORTTRIG4 – Observation Architecture Report Trigger

OAREPORTRIG4— Observation Architecture Report Trigger					
Register Type:	MMIO				
Address Offset:	2370h				
Project:	All				
Default Value:	00000000h				
Access:	RW				
Size (in bits):	32				
This register is used to program the OA unit.					
Bit De	**scription**				
31:28	**NOA Signal Select 7** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
27:24	**NOA Signal Select 6** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
23:20	**NOA Signal Select 5** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
19:16	**NOA Signal Select 4** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
15:12	**NOA Signal Select 3** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
11:8	**NOA Signal Select 2** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
7:4	**NOA Signal Select 1** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4
3:0	**NOA Signal Select 0** Select 1 of the 16 input NOA signals	Project:	All	Format:	U4

1.1.14.13 CEC0-0 – Customizable Event Creation

CEC0-0— Customizable Event Creation

Register Type: MMIO
Address Offset: 2390h
Project: All
Default Value: 00000000h
Access: Write Only
Size (in bits): 32

This register is used to program the OA unit.

Bit De	scription
31:21	**Reserved** Project: [DevSNB] Format: MBZ
20:19	**Clock Domain** Project: DevSNB Format: U2 Selects clock domains for DELAY flops and BOOLEAN EVENT flops. The encoding of this field is device specific.

Value Na	me	Description	Project
000b	crclk		All
001b	Reserved		All
010b	hclk		All
011b	Reserved		All
100b	mcclk		All
101b	Reserved		All
110b	lgclk		All
111b	Reserved		All

20:19	**Reserved** Project: Format: MBZ
18:3	**Compare Value** Project: All Format: U16 Bit field LSB corresponds to NOA bit 0. This field is loaded to compare against the 8 NOA signals that are fed into this block. The type of comparison that is done is controlled by the Compare Function. When the compare function is true, then the signal for the NOA event is asserted. This in turn can be counted by any of the CHAP counters.

CEC0-0— Customizable Event Creation

2:0 **Compare Function** Project: All Format: U3

Value Name		Description	Project
000b	Any Are Equal	Compare and assert if any are equal (Can be used as OR function)	All
001b	Greater Than	Compare and output signal if greater than	All
010b	Equal	Compare and assert output if equal to (Can also be used as AND function)	All
011b	Greater Than or Equal	Compare and assert output if greater than or equal	All
100b	Less Than	Compare and assert output if less than	All
101b	Not Equal	Compare and assert output if not equal	All
110b	Less Than or Equal	Compare and assert output if less than or equal	All
111b	Reserved		All

1.1.14.14 CEC0-1 – Customizable Event Creation

CEC0-1— Customizable Event Creation	
Register Type:	MMIO
Address Offset:	2394h
Project:	All
Default Value:	00000000h
Access:	Write Only
Size (in bits):	32
This register is used to program the OA unit.	
Bit De	**scription**
31:16	**Considerations** Project: All Format: U32 Bit field LSB corresponds to NOA bit 0. 0: The NOA bit is considered in event calculations. 1: The NOA bit is delayed by 1 clock before considering it in event calculations. This is particularly useful for doing state machine arc coverage. For example, NOA bits 3:0 and NOA 7:4 could be programmed to the same 4 present state, state machine signals. The appropriate inversion selections would be made depending on which state transition is of interest. Bits 31:28 in the delay selection would be programmed to "1111", indicating use a pipe delayed version of the state signals. The resulting "AND" of the now preconditioned NOA 7:4 and NOA 3:0 signals would indicate the number of times the arc of interest was taken. This could be recorded with the CHAP counters.
15:0	**Mask** Project: All Format: U32 Bit field LSB corresponds to NOA bit 0. These 8 bits are used to mask off entries from the comparison. For each bit: 0: This NOA bit is considered in event calculations. 1: This NOA bit is ignored in event calculations.

1.1.14.15 CEC1-0 – Customizable Event Creation

CEC1-0— Customizable Event Creation				
Register Type: MMIO **Address Offset:** 2398h **Project:** All **Default Value:** 00000000h **Access:** Write Only **Size (in bits):** 32				
This register is used to program the OA unit.				
Bit De	**scription**			
31:21	**Reserved** Project: All Format: MBZ			
20:19	**Clock Domain** Project: [DevSNB] Format: U2 Selects clock domains for DELAY flops and BOOLEAN EVENT flops. The encoding of this field is device specific.			
	Value Na	**me**	**Description**	**Project**
	000b	crclk		All
	001b	Reserved		All
	010b	hclk		All
	011b	Reserved		All
	100b	mcclk		All
	101b	Reserved		All
	110b	lgclk		All
	111b	Reserved		All
20:19	**Reserved** Project: Format: MBZ			
18:3	**Compare Value** Project: All Format: U16 Bit field LSB corresponds to NOA bit 0. This field is loaded to compare against the 8 NOA signals that are fed into this block. The type of comparison that is done is controlled by the Compare Function. When the compare function is true, then the signal for the NOA event is asserted. This in turn can be counted by any of the CHAP counters.			

CEC1-0— Customizable Event Creation	
2:0	**Compare Function** Project: All Format: U3

Value Name	Description	Project
000b	Any Are Equal	Compare and assert if any are equal (Can be used as OR function)	All
001b	Greater Than	Compare and output signal if greater than	All
010b	Equal	Compare and assert output if equal to (Can also be used as AND function)	All
011b	Greater Than or Equal	Compare and assert output if greater than or equal	All
100b	Less Than	Compare and assert output if less than	All
101b	Not Equal	Compare and assert output if not equal	All
110b	Less Than or Equal	Compare and assert output if less than or equal	All
111b	Reserved		All

1.1.14.16 CEC1-1 – Customizable Event Creation

CEC1-1— Customizable Event Creation	
Register Type:	MMIO
Address Offset:	239Ch
Project:	All
Default Value:	00000000h
Access:	Write Only
Size (in bits):	32
This register is used to program the OA unit.	

Bit Description	
31:16	**Considerations** Project: All Format: U32 Bit field LSB corresponds to NOA bit 0. 0: The NOA bit is considered in event calculations. 1: The NOA bit is delayed by 1 clock before considering it in event calculations. This is particularly useful for doing state machine arc coverage. For example, NOA bits 3:0 and NOA 7:4 could be programmed to the same 4 present state, state machine signals. The appropriate inversion selections would be made depending on which state transition is of interest. Bits 31:28 in the delay selection would be programmed to "1111", indicating use a pipe delayed version of the state signals. The resulting "AND" of the now preconditioned NOA 7:4 and NOA 3:0 signals would indicate the number of times the arc of interest was taken. This could be recorded with the CHAP counters.
15:0	**Mask** Project: All Format: U32 Bit field LSB corresponds to NOA bit 0. These 8 bits are used to mask off entries from the comparison. For each bit: 0: This NOA bit is considered in event calculations. 1: This NOA bit is ignored in event calculations.

1.1.14.17 CEC2-0 – Customizable Event Creation

CEC2-0— Customizable Event Creation				
Register Type: MMIO **Address Offset:** 23A0h **Project:** All **Default Value:** 00000000h **Access:** Write Only **Size (in bits):** 32				
This register is used to program the OA unit.				
Bit De	**scription**			
31:21	**Reserved** Project: All Format: MBZ			
20:19	**Clock Domain** Project: [DevSNB] Format: U2 Selects clock domains for DELAY flops and BOOLEAN EVENT flops. The encoding of this field is device specific.			
	Value Na	**me**	**Description**	**Project**
	000b	crclk		All
	001b	Reserved		All
	010b	hclk		All
	011b	Reserved		All
	100b	mcclk		All
	101b	Reserved		All
	110b	lgclk		All
	111b	Reserved		All
20:19	**Reserved** Project: Format: MBZ			
18:3	**Compare Value** Project: All Format: U16 Bit field LSB corresponds to NOA bit 0. This field is loaded to compare against the 8 NOA signals that are fed into this block. The type of comparison that is done is controlled by the Compare Function. When the compare function is true, then the signal for the NOA event is asserted. This in turn can be counted by any of the CHAP counters.			

CEC2-0— Customizable Event Creation	
2:0	**Compare Function** Project: All Format: U3

Value Name	Description	Project
000b	Any Are Equal	Compare and assert if any are equal (Can be used as OR function)	All
001b	Greater Than	Compare and output signal if greater than	All
010b	Equal	Compare and assert output if equal to (Can also be used as AND function)	All
011b	Greater Than or Equal	Compare and assert output if greater than or equal	All
100b	Less Than	Compare and assert output if less than	All
101b	Not Equal	Compare and assert output if not equal	All
110b	Less Than or Equal	Compare and assert output if less than or equal	All
111b	Reserved		All

1.1.14.18 CEC2-1 – Customizable Event Creation

CEC2-1— Customizable Event Creation	
Register Type:	MMIO
Address Offset:	23A4h
Project:	All
Default Value:	00000000h
Access:	Write Only
Size (in bits):	32
This register is used to program the OA unit.	

Bit	Description
31:16	**Considerations** Project: All Format: U32 Bit field LSB corresponds to NOA bit 0. 0: The NOA bit is considered in event calculations. 1: The NOA bit is delayed by 1 clock before considering it in event calculations. This is particularly useful for doing state machine arc coverage. For example, NOA bits 3:0 and NOA 7:4 could be programmed to the same 4 present state, state machine signals. The appropriate inversion selections would be made depending on which state transition is of interest. Bits 31:28 in the delay selection would be programmed to "1111", indicating use a pipe delayed version of the state signals. The resulting "AND" of the now preconditioned NOA 7:4 and NOA 3:0 signals would indicate the number of times the arc of interest was taken. This could be recorded with the CHAP counters.
15:0	**Mask** Project: All Format: U32 Bit field LSB corresponds to NOA bit 0. These 8 bits are used to mask off entries from the comparison. For each bit: 0: This NOA bit is considered in event calculations. 1: This NOA bit is ignored in event calculations.

1.1.14.19 CEC3-0 – Customizable Event Creation

CEC3-0— Customizable Event Creation

Register Type: MMIO
Address Offset: 23A8h
Project: All
Default Value: 00000000h
Access: Write Only
Size (in bits): 32

This register is used to program the OA unit.

Bit De	scription
31:21	**Reserved** Project: All Format: MBZ
20:19	**Clock Domain** Project: All Format: U2 Selects clock domains for DELAY flops and BOOLEAN EVENT flops. The encoding of this field is device specific.

Value Na	me	Description	Project
000b	crclk		All
001b	Reserved		All
010b	hclk		All
011b	Reserved		All
100b	mcclk		All
101b	Reserved		All
110b	lgclk		All
111b	Reserved		All

20:19	**Reserved** Project: Format: MBZ
18:3	**Compare Value** Project: All Format: U16 Bit field LSB corresponds to NOA bit 0. This field is loaded to compare against the 8 NOA signals that are fed into this block. The type of comparison that is done is controlled by the Compare Function. When the compare function is true, then the signal for the NOA event is asserted. This in turn can be counted by any of the CHAP counters.

CEC3-0— Customizable Event Creation

2:0 **Compare Function** Project: All Format: U3

Value Name		Description	Project
000b	Any Are Equal	Compare and assert if any are equal (Can be used as OR function)	All
001b	Greater Than	Compare and output signal if greater than	All
010b	Equal	Compare and assert output if equal to (Can also be used as AND function)	All
011b	Greater Than or Equal	Compare and assert output if greater than or equal	All
100b	Less Than	Compare and assert output if less than	All
101b	Not Equal	Compare and assert output if not equal	All
110b	Less Than or Equal	Compare and assert output if less than or equal	All
111b	Reserved		All

1.1.14.20 CEC3-1 – Customizable Event Creation

CEC3-1— Customizable Event Creation	
Register Type:	MMIO
Address Offset:	23ACh
Project:	All
Default Value:	00000000h
Access:	Write Only
Size (in bits):	32
This register is used to program the OA unit.	
Bit De	**scription**
31:16	**Considerations** Project: All Format: U32 Bit field LSB corresponds to NOA bit 0. 0: The NOA bit is considered in event calculations. 1: The NOA bit is delayed by 1 clock before considering it in event calculations. This is particularly useful for doing state machine arc coverage. For example, NOA bits 3:0 and NOA 7:4 could be programmed to the same 4 present state, state machine signals. The appropriate inversion selections would be made depending on which state transition is of interest. Bits 31:28 in the delay selection would be programmed to "1111", indicating use a pipe delayed version of the state signals. The resulting "AND" of the now preconditioned NOA 7:4 and NOA 3:0 signals would indicate the number of times the arc of interest was taken. This could be recorded with the CHAP counters.
15:0	**Mask** Project: All Format: U32 Bit field LSB corresponds to NOA bit 0. These 8 bits are used to mask off entries from the comparison. For each bit: 0: This NOA bit is considered in event calculations. 1: This NOA bit is ignored in event calculations.

1.1.14.21 OANOASELECT – Observation Architecture NOA select [DevSNB]

OANOASELECT— Observation Architecture NOA Select

Register Type: MMIO
Address Offset: 236Ch
Project: All
Default Value: 00000000h
Access: RW
Size (in bits): 32

This register is used to program the OA unit.

Bit De	scription
31:0	**Rerserved** Project: All

Value Na	me	Description	Project
00b	csclk	NOA FM CS clk	All
01b	crclk	NOA FM CR clk	All
10b	crmclk	NOA FM CRM clk	All
11b	Reserved		All

29:28 **NOA Select Bits for Counter 14** Project: All

Value Na	me	Description	Project
00b	csclk	NOA FM CS clk	All
01b	crclk	NOA FM CR clk	All
10b	crmclk	NOA FM CRM clk	All
11b	Reserved		All

27:26 **NOA Select Bits for Counter 13** Project: All

Value Na	me	Description	Project
00b	csclk	NOA FM CS clk	All
01b	crclk	NOA FM CR clk	All
10b	crmclk	NOA FM CRM clk	All
11b	Reserved		All

OANOASELECT— Observation Architecture NOA Select

25:24 **NOA Select Bits for Counter 12** Project: All

Value Na	me	Description	Project
00b	csclk	NOA FM CS clk	All
01b	crclk	NOA FM CR clk	All
10b	crmclk	NOA FM CRM clk	All
11b	Reserved		All

23:22 **NOA Select Bits for Counter 11** Project: All

Value Na	me	Description	Project
00b	csclk	NOA FM CS clk	All
01b	crclk	NOA FM CR clk	All
10b	crmclk	NOA FM CRM clk	All
11b	Reserved		All

21:20 **NOA Select Bits for Counter 10** Project: All

Value Na	me	Description	Project
00b	csclk	NOA FM CS clk	All
01b	crclk	NOA FM CR clk	All
10b	crmclk	NOA FM CRM clk	All
11b	Reserved		All

19:18 **NOA Select Bits for Counter 9** Project: All

Value Na	me	Description	Project
00b	csclk	NOA FM CS clk	All
01b	crclk	NOA FM CR clk	All
10b	crmclk	NOA FM CRM clk	All
11b	Reserved		All

OANOASELECT— Observation Architecture NOA Select

17:16 **NOA Select Bits for Counter 8** Project: All

Value Na	me	Description	Project
00b	csclk	NOA FM CS clk	All
01b	crclk	NOA FM CR clk	All
10b	crmclk	NOA FM CRM clk	All
11b	Reserved		All

15:14 **NOA Select Bits for Counter 7** Project: All

Value Na	me	Description	Project
00b	csclk	NOA FM CS clk	All
01b	crclk	NOA FM CR clk	All
10b	crmclk	NOA FM CRM clk	All
11b	Reserved		All

13:12 **NOA Select Bits for Counter 6** Project: All

Value Na	me	Description	Project
00b	csclk	NOA FM CS clk	All
01b	crclk	NOA FM CR clk	All
10b	crmclk	NOA FM CRM clk	All
11b	Reserved		All

11:10 **NOA Select Bits for Counter 5** Project: All

Value Na	me	Description	Project
00b	csclk	NOA FM CS clk	All
01b	crclk	NOA FM CR clk	All
10b	crmclk	NOA FM CRM clk	All
11b	Reserved		All

OANOASELECT— Observation Architecture NOA Select

9:8 **NOA Select Bits for Counter 4** Project: All

Value Na	me	Description	Project
00b	csclk	NOA FM CS clk	All
01b	crclk	NOA FM CR clk	All
10b	crmclk	NOA FM CRM clk	All
11b	Reserved		All

7:6 **NOA Select Bits for Counter 3** Project: All

Value Na	me	Description	Project
00b	csclk	NOA FM CS clk	All
01b	crclk	NOA FM CR clk	All
10b	crmclk	NOA FM CRM clk	All
11b	Reserved		All

5:4 **NOA Select Bits for Counter 2** Project: All

Value Na	me	Description	Project
00b	csclk	NOA FM CS clk	All
01b	crclk	NOA FM CR clk	All
10b	crmclk	NOA FM CRM clk	All
11b	Reserved		All

3:2 **NOA Select Bits for Counter 1** Project: All

Value Na	me	Description	Project
00b	csclk	NOA FM CS clk	All
01b	crclk	NOA FM CR clk	All
10b	crmclk	NOA FM CRM clk	All
11b	Reserved		All

OANOASELECT— Observation Architecture NOA Select

1:0 **NOA Select Bits for Counter 0** Project: All

Value Na	me	Description	Project
00b	csclk	NOA FM CS clk	All
01b	crclk	NOA FM CR clk	All
10b	crmclk	NOA FM CRM clk	All
11b	Reserved		All

1.1.15 Frame Buffer Compression Control ([DevCL] Only)

This section describes the registers associated with the Frame Buffer Compression function. The primary motivation of FBC is power savings and thus it is only applicable to the Mobile Product.

Programming Notes:

- Frame buffer compression has to be disabled (via FBC_CONTROL[31] = 0), and software has to wait until compression not in progress (FBC_STATUS[31] == 0) before changing any of the following fields:
  - FBC_CFB_BASE
  - FBC_LL_BASE
  - FBC_CONTROL[Mode Select]
  - FBC_CONTROL[Compressed Frame Buffer Stride]
  - FBC_CONTROL[Fence Number]

1.1.15.1 FBC_CFB_BASE — Compressed Frame Buffer Base Address

FBC_CFB_BASE — Compressed Frame Buffer Base Address	
Register Type:	MMIO
Address Offset:	3200h
Project:	DevCL
Default Value:	0000 0000h
Access:	R/W
Size (in bits):	32
This register specifies the physical memory address at which the Compressed Frame Buffer is located. **Note that the Compressed Frame Buffers must be in Non Cacheable memory and not relocated while FBC is active.**	

Bit De	scription
31:12	**Compressed Frame Buffer Address** Project: DevCL Default Value: 0h Address: PhysicalAddress[31:12] This register specifies Bits 31:12 of the physical address of the Compressed Frame Buffer. **Programming Notes** Software must guarantee that the Compressed Frame Buffer is stored in contiguous physical memory. The buffer must be 4K byte aligned. This field should not be changed unless FBC is inactive (the first VBlank start after **Enable Frame Buffer Compression** has been cleared.)
11:0	**Reserved** Project: DevCL Format: MBZ

1.1.15.2 FBC_LL_BASE — Compressed Frame Line Length Buffer Address

FBC_LL_BASE — Compressed Frame Line Length Buffer Address	
Register Type:	MMIO
Address Offset:	3204h
Project:	DevCL
Default Value:	0000 0000h
Access:	R/W
Size (in bits):	32
This register specifies the physical memory address at which the Compressed Frame Line Length Buffer is located. **Note that the Compressed Frame Buffers must be in Non Cacheable memory and not relocated while FBC is active.**	

Bit De	scription
31:12	**Compressed Frame Line Length Buffer Address** Project: DevCL Default Value: 0h Address: PhysicalAddress[31:12] This register specifies Bits 31:12 of the physical address of the Compressed Frame Line Length Buffer. **Programming Notes** Software must guarantee that the Compressed Frame Line Length Buffer is stored in contiguous physical memory. The buffer must be 4K byte aligned. This field should not be changed unless FBC is inactive (the first VBlank start after **Enable Frame Buffer Compression** has been cleared.)
11:0	**Reserved** Project: DevCL Format: MBZ

1.1.15.3 FBC_CONTROL — Frame Buffer Compression Control Register

FBC_CONTROL — Frame Buffer Compression Control Register

Register Type: MMIO
Address Offset: 3208h
Project: DevCL
Default Value: 0000 0000h
Access: R/W
Size (in bits): 32

This register is used to control the operation of RLE-FBC.

Bit De	scription
31	**Enable Frame Buffer Compression** Project: DevCL Default Value: 0h Format: Enable This bit is used to globally enable or disable the RLE-FBC function (compression and decompression) at the next VBlank start.

Value Na	me	Description	Project
0h	Disable	Disable frame buffer compression.	DevCL
1h	Enable	Enable frame buffer compression.	DevCL

Bit De	scription
30	**Mode Select** Project: DevCL Default Value: 0h Format: U1

Value Na	me	Description	Project
0h	Single Pass	Single Pass mode	DevCL
1h	Periodic Pass	Periodic mode	DevCL

Bit De	scription
29:16	**Interval** Project: DevCL Default Value: 0h Format: U14 Range [1,16383] This is interval for which the compressor waits between passes. In Periodic Mode this field determines the interval length, in terms of frames (VBlanks). Zero is an illegal value.

FBC_CONTROL — Frame Buffer Compression Control Register	
15	**Stop Compressing on Modification (DEBUG ONLY)** Project: DevCL Format: Enable If set to '1' the compressor will abort a subsequent compressing pass when any modification to the source frame buffer is detected.
14	**Uncompressible Enable** Project: DevCL Format: Enable If set to a '1' the compressor marks as "Uncompressible 10" (see the FBC_TAG register) if any scanline in a pair cannot be compressed. In Default mode Uncompressible mode is turned off.
13	**Reserved** Project: DevCL Format: MBZ
12:5	**Compressed Frame Buffer Stride** Project: DevCL Format: (Stride in 64Byte units) – 1 This is the stride for the compressed frame buffer. This value is used to determine the line-to-line increment for the compressed frame buffer. Lines that cannot be compressed to a stride size or less are not compressed at all. This field must be set to a value less than or equal to the stride of the source (uncompressed) frame buffer. 00h = 64B stride
4	**Reserved** Project: DevCL Format: MBZ
3:0	**Fence Number** Project: DevCL Format: U3 This field specifies the FENCE number corresponding to the placement of the uncompressed frame buffer. (Note that only tiled frame buffers can be compressed). This field is double buffered in hardware. Only the host accesses the uncompressed frame buffer using a fence.

1.1.15.4 FBC_COMMAND — Frame Buffer Compression Command Register

FBC_COMMAND — Frame Buffer Compression Command Register	
Register Type:	MMIO
Address Offset:	320Ch
Project:	DevCL
Default Value:	0000 0000h
Access:	R/W
Size (in bits):	32
This register is used to request a frame buffer compression pass while in Single Pass mode.	
Bit De	**scription**
31:1	**Reserved** Project: DevCL Format: MBZ
0	**Compress Enable** Project: DevCL Format: Enable Software can set this bit to trigger compression in Single Pass mode. The compressor clears this bit after the compression pass is completed. This bit is ignored in Periodic Mode (i.e., it will not cause a compression pass and will always read as '0').

1.1.15.5 FBC_STATUS — Frame Buffer Compression Status Register

FBC_STATUS — Frame Buffer Compression Status Register	
Register Type:	MMIO
Address Offset:	3210h
Project:	DevCL
Default Value:	2000 0000h
Access:	RO / R/W
Size (in bits):	32
This register contains status information associated with the RLE-FBC function. The information is read-only in normal operation, though some fields can be programmed as a TEST MODE.	

Bit De	scription
31	**Compressing** Project: DevCL Security: RO Default Value: 0h Format: Flag This status bit indicates that the device is currently within a compression pass.
30	**Compressed** Project: DevCL Security: RO normally, R/W TEST MODE Default Value: 0h Format: Flag This bit indicates that a compressed frame buffer is available at the address contained in the FB_CFB_BASE register. In normal operation the compressor sets this bit when it has completed the compression pass. During compression this bit is not set. As a test mode this bit can be set if there is a software-created compressed buffer available at the address in the FB_CFB_BASE register. Test-Mode software must check that compression is **not** in progress before setting this bit. If RLE-FBC is enabled, the compressor will clear this bit when it starts the next recompression pass.

FBC_STATUS — Frame Buffer Compression Status Register	
29	**Any Modified** Project: DevCL Security: RO normally, R/W TEST MODE Default Value: 1h Format: Flag 1 = (default) Indicates that the frame buffer has been modified since the last compression pass. The compressor sets this bit on the first write to the frame buffer from the application/driver or upon an allocation within the render cache (e.g., as a result of Blt, 3D or MPEG activity). The fence number and frame buffer base address are used to determine if a write modified the frame buffer. The bit is cleared by the compressor at the start of the next compression pass. In normal operation this bit is read only (software must not write this bit) and defaults to a **"1"**. As a test mode this bit can be set if there is a software-created compressed buffer with modified lines available at the address contained the FB_CFB_BASE register. SW must check that compression is **not** in progress before setting this bit. If enabled, the compressor will clear this bit when it initiates the next compression pass. This test mode is used for continuous-mode compression testing.
28:11	**Reserved** Project: DevCL Format: MBZ
10:0	**Current Line Compressing** Project: DevCL Security: RO Default Value: 0h Format: U11 This read only field indicates the line number that the compressor is currently processing. If this field is 0 and the **Compressing** bit (Bit 31) is set, the compressor is currently on display frame line 1.

1.1.15.6 FBC_CONTROL2— Frame Buffer Compression 2nd Control Register

FBC_CONTROL2— Frame Buffer Compression 2nd Control Register	
Register Type:	MMIO
Address Offset:	3214h
Project:	DevCL
Default Value:	0000 0000h
Access:	R/W
Size (in bits):	32
This register is used to control the operation of RLE-FBC.	
Bit De	**scription**
31:3	**Reserved** Project: DevCL Format: MBZ

FBC_CONTROL2— Frame Buffer Compression 2nd Control Register

4

Double Buffer FBC Fence and Fence_DisplayY Offset Register Fields

Project: DevCL

Default Value: 0h

Format: Disable

Value Na	me	Description	Project
0h		Double buffer	DevCL
1h		Don't double buffer	DevCL

3:2

FBC C3 Mode

Project: DevCL

Default Value: 0h

Format: U2

Value Na	me	Description	Project
00		FBC IDLENESS is not looked at in order to enter Self Refresh	DevCL
01		FBC IDLENESS is looked at in order to enter Self Refresh	DevCL
10		FBC IDLENESS is looked at in order to enter Self Refresh. But FBC enters IDLE as it finishes compressing the current scanline pair and enters IDLE as soon as csunit asserts the inc3 signal.	DevCL
11	Reserved	Reserved	DevCL

1

CPU Fence enable

Project: DevCL

Default Value: 0h

Format: Enable

Value Na	me	Description	Project
0h		Display Buffer is not in a CPU fence. No modifications are expected from CPU to the Display Buffer.	DevCL
1h		Display Buffer exists in a CPU fence.	DevCL

FBC_CONTROL2— Frame Buffer Compression 2nd Control Register	
0	**Frame Buffer Compression Display Plane Select A/B** Project: DevCL Default Value: 0h Format: Flag

Value Na	me	Description	Project
0h		Enable frame buffer compression on Plane A.	All
1h		Enable frame buffer compression on Plane B.	All

Programming Notes	Project
Before changing this bit s/w needs to make sure that FBC is disabled and the “COMPRESSING” bit in the FBC_CONTROL register comes to a “0”.	DevCL

1.1.15.7 FBC_DISPYOFF — FBC Fence Display Buffer Y offset

FBC_DISPYOFF — FBC Fence Display Buffer Y offset	
Register Type:	MMIO
Address Offset:	321Bh
Project:	DevCL
Default Value:	0000 0000h
Access:	R/W
Size (in bits):	32

Desc

Bit De	scription
31:12	**Reserved** Project: DevCL Format: MBZ
11:0	**Fence_YDisp** Project: DevCL Format: U12 Y offset from the fence to the Display Buffer base

1.1.15.8 FBC_MOD_NUM— FBC Number of Modifications for Recompression

FBC_MOD_NUM— FBC Number of Modifications for Recompression					
Register Type:	MMIO				
Address Offset:	3220h				
Project:	DevCL				
Default Value:	0000 0000h				
Access:	R/W				
Size (in bits):	32				
Trusted Type:	1				
The purpose of this register is to avoid SR exit unless the programmed number of modifications have been made to the Display buffer.					
Bit De	**scription**				
31:1	**FBC_Mod_Num**	Project:	DevCL	Format:	U12
	Number of modifications to the display buffer required before recompression is attempted. If the number of modifications to the Frame Buffer is not equal to the programmed count value at the end of the interval, re-compression is not attempted.				
0	**FBC_Mod_Num_Valid**	Project:	DevCL	Format:	Flag
	Only if this bit is set will the above count value be looked at.				

1.1.15.9 FBC_TAG — Frame Buffer Compression TAG Interface (DEBUG)

FBC_TAG — Frame Buffer Compression TAG Interface (DEBUG)	
Register Type:	MMIO
Address Offset:	3300h
Project:	All
Default Value:	00000000h;
Access:	R/W
Size (in bits):	49x32
Trusted Type:	1

The device implements 49 DWords of Tag data for RLE-FBC compression. Each DWord contains storage for a 2-bit Tag value associated with a frame buffer line pair.

49 DWords are required to support the required 1536 display lines (= 48 x 32), as an extra DWord may be required due to the alignment of the source (uncompressed) frame buffer. I.e., if the source frame buffer starts on an odd tile line, line 0 corresponds to bit 1 of 3300 (bit 0 is unused) and the 49th DWord may be required. If the source frame buffer starts on an even tile line, line 0 corresponds to bit 0 of 3300.

FBC_TAG — Frame Buffer Compression TAG Interface (DEBUG)		
DWord	**Bit**	**Description**
0..48	31:30	**Tag for lines 30&31** Project: All Format: FBC Tag For lines: (DWord) + 30 and (DWord) + 31
	29:28	**Tag for lines 29&28** Project: All Format: FBC Tag For lines: (DWord) + 30 and (DWord) + 31
	27:26	**Tag for lines 27&26** Project: All Format: FBC Tag For lines: (DWord) + 30 and (DWord) + 31
	25:24	**Tag for lines 25&24** Project: All Format: FBC Tag For lines: (DWord) + 30 and (DWord) + 31
	23:22	**Tag for lines 23&22** Project: All Format: FBC Tag For lines: (DWord) + 30 and (DWord) + 31
	21:20	**Tag for lines 21&20** Project: All Format: FBC Tag For lines: (DWord) + 30 and (DWord) + 31
	19:18	**Tag for lines 19&18** Project: All Format: FBC Tag For lines: (DWord) + 30 and (DWord) + 31
	17:16	**Tag for lines 17&16** Project: All Format: FBC Tag For lines: (DWord) + 30 and (DWord) + 31
	15:14	**Tag for lines 15&14** Project: All Format: FBC Tag For lines: (DWord) + 30 and (DWord) + 31
	13:12	**Tag for lines 13&12** Project: All Format: FBC Tag For lines: (DWord) + 30 and (DWord) + 31
	11:10	**Tag for lines 11&10** Project: All Format: FBC Tag For lines: (DWord) + 30 and (DWord) + 31
	9:8	**Tag for lines 9&8** Project: All Format: FBC Tag For lines: (DWord) + 30 and (DWord) + 31
	7:6	**Tag for lines 7&6** Project: All Format: FBC Tag For lines: (DWord) + 30 and (DWord) + 31
	5:4	**Tag for lines 5&4** Project: All Format: FBC Tag For lines: (DWord) + 30 and (DWord) + 31
	3:2	**Tag for lines 3&2** Project: All Format: FBC Tag For lines: (DWord) + 30 and (DWord) + 31
	1:0	**Tag for lines 1&0** Project: All Format: FBC Tag For lines: (DWord) + 30 and (DWord) + 31

FBC_TAG — Frame Buffer Compression TAG Interface (DEBUG)

	31:0	**Tag for lines DW# + 1&0** Project: All Format: FBC Tag See below For lines: (DWord) + 30 and (DWord) + 31

Value Name	Description	Project
00 Modified	At least one of the associated lines was modified since the last compression pass started.	All
01 Uncompressed	The associated lines are uncompressed and are candidate for compression in the next pass	All
10 Uncompressible	The associated lines are uncompressible and are not candidate for compression in the next pass.	All
11 Compressed	The associated lines are compressed	All

1.2 Fence Registers

1.2.1 FENCE — Graphics Memory Fence Table Registers

FENCE — Graphics Memory Fence Table Registers	
Register Type:	MMIO
Address Offset:	3000h
Project:	All
Default Value:	00000000h;
Access:	R/W
Size (in bits):	16x64
Trusted Type:	1

Address Offset:

03000h – 03007h: FENCE_0
:
:
0307Ch – 0307Fh: FENCE_15

The graphics device performs address translation from linear space to tiled space for a CPU access to graphics memory (See *Memory Interface Functions* chapter for information on these memory layouts) using the fence registers. Note that the fence registers are used **only for CPU accesses to gfx memory**. Graphics rendering/display pipelines use Per Surface Tiling (PST) parameters (found in SURFACE_STATE – see the *Sampling Engine* chapter) to access tiled gfx memory.

The intent of tiling is to locate graphics data that are close (in X and Y surface axes) in one physical memory page while still locating some amount of line oriented data sequentially in memory for display efficiency. All 3D rendering is done such that the QWords of any one span are all located in the same memory page, improving rendering performance. Applications view surfaces as linear, hence when the cpu access a surface that is tiled, the gfx hardware must perform linear to tiled address conversion and access the correct physical memory location(s) to get the data.

Tiled memory is supported for rendering and display surfaces located in graphics memory. A tiled memory surface is a surface that has a width and height that are subsets of the tiled region's pitch and height. The device maintains the constants required by the memory interface to perform the address translations. Each tiled region can have a different pitch and size. The CPU-memory interface needs the surface pitch and tile height to perform the address translation. It uses the GMAddr (PCI-BAR) offset address to compare with the fence start and end address, to determine if the rendering surface is tiled. The tiled address is generated based on the tile orientation determined from the matching fence register. Fence ranges are at least 4 KB aligned. Note that the fence registers are used only for CPU accesses to graphics memory.

A Tile represents 4 KB of memory. Tile height is 8 rows for X major tiles and 32 rows for Y major tiles. Tile Pitch is 512Bs for X major tiles and 128Bs for Y major tiles. The surface pitch is programmed in 128B units such that the pitch is an integer multiple of "tile pitch".

Engine restrictions on tile surface usage are detailed in Surface Placement Restrictions (Memory Interface Functions). Note that X major tiles can be used for Sampler, Color, Depth, motion compensation references and motion compensation destination, Display, Overlay, GDI Blt source and destination surfaces. Y major tiles can be used for Sampler, depth, color and motion compensation assuming they do not need to be displayed. GDI Blit operations, overlay and display cannot used Tiled Y orientations.

A "PST" graphics surface that will also be accessed via fence needs its base address to be tile row aligned.

FENCE — Graphics Memory Fence Table Registers

Hardware handles the flushing of any pending cycles when software changes the fence upper/lower bounds.

Fence Table Registers occupy the address range specified above. Each Fence Table Register has the following format.

FENCE registers are *not* reset by a graphics reset. They will maintain their values unless a full chipset reset is performed.

DWord Bit		Description
0..15	63:44	**Fence Upper Bound** Project: All Address: GraphicsAddress[31:12] Bits 31:12 of the ending Graphics Address of the fence region. Fence regions must be aligned to a 4KB page. This address represents the last 4KB page of the fence region (Upper Bound is included in the fence region). Graphics Address is the offset within GMADR space.
	45:32	**Reserved** Project: All Format: MBZ
	31:12	**Fence Lower Bound** Project: All Address: GraphicsAddress[31:12] Bits 31:12 of the starting Graphics Address of the fence region. Fence regions must be aligned to 4KB. This address represents the first 4KB page of the fence region (Lowe Bound is included in the fence region). Graphics Address is the offset within GMADR space.
	11:2	**Fence Pitch** Project: All Default Value: 0h DefaultVaueDesc Format: U10-1 Width in 128 bytes This field specifies the width (pitch) of the fence region in multiple of “tile width”. For Tile X this field must be programmed to a multiple of 512B (“003” is the minimum value) and for Tile Y this field must be programmed to a multiple of 128B (“000” is the minimum value). 000h = 128B 001h = 256B ... 3FFh = 128KB

FENCE — Graphics Memory Fence Table Registers

	1	**Tile Walk** Project: All Format: MI_TileWalk This field specifies the spatial ordering of QWords within tiles.

Value Name	Description	Project
0h	MI_TILE_XMAJOR Consecutive SWords (32 Bytes) sequenced in the X direction	All
1h	MI_TILE_YMAJOR Consecutive OWords (16 Bytes) sequenced in the Y direction	All

0	**Fence Valid** Project: All Format: MI_ FenceValid This field specifies whether or not this fence register defines a fence region.

Value Name	Description	Project
0h	MI_FENCE_INVALID	All
1h	MI_FENCE_VALID	All

1.3 Memory Interface Commands for Rendering Engine

1.3.1 Introduct ion

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 original graphics processing engine. The term “for Rendering Engine” in the title has been added to differentiate this chapter from a similar one describing the MI commands for the Media Decode Engine.

The commands detailed in this chapter are used across products within the Gen4+ family. However, slight changes may be present in some commands (i.e., for features added or removed), or some commands may be removed entirely.

1.3.2 Software Synchronization Commands

To support mid-triangle interruption, certain commands need to be placed in a temporary location in hardware until primitive commands are complete. This introduces out-of-order command execution. Below show the commands that are affected. Note that the INSTPM register has a bit that is used to force in-order execution.

Command	Qualifications
MI_NOOP	When writing to the NOOPID register
MI_USER_INTERRUPT	Always
MI_PROBE	Writing out new value after check
MI_UNPROBE	Always
MI_SEMAPHORE_MBOX	Memory write
MI_STORE_DATA_IMM	Always
MI_STORE_DATA_INDEX	Always
MI_LOAD_REGISTER_IMM	Always
MI_UPDATE_GTT	Always
MI_STORE_REGISTER_MEM	Register read is done in-order, register write done out-of-order

1.3.3 MI_ARB_CHECK

MI_ARB_CHECK			
Project:	All	**Length Bias:**	1
Engine:	**Render**		

The MI_ARB_CHECK instruction is used to check the ring buffer double buffered head pointer (register UHPTR). This instruction can be used to pre-empt the current execution of the ring buffer. Note that the valid bit in the updated head pointer register needs to be set for the command streamer to be pre-empted.

Programming Note:

- The current head pointer is loaded with the updated head pointer register independent of the location of the updated head
- If the current head pointer and the updated head pointer register are equal, hardware will automatically reset the valid bit corresponding to the UHPTR
- For Gen6 this instruction can be placed only in a ring buffer, never in a batch buffer. For Gen7+ it can be in either a ring buffer or batch buffer.
- For pre-emption, the wrap count in the ring buffer head register is no longer maintained by hardware. The hardware updates the wrap count to the value in the UHPTR register.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 05h MI_ARB_CHECK Format: OpCode
	22:0	**Reserved** Project: All Format: MBZ

1.3.4 MI_BATCH_BUFFER_END

MI_BATCH_BUFFER_END			
Project:	All	**Length Bias:**	1
Engine:	**Render**		

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	31:0	**Semaphore 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	**Semaphore Address** Qword address to fetch Data Dword(DW0) from memory. HW will compare the Data Dword(DW0) with Semaphore Data Dword
	2:0	**Reserved** Project: All Format: MBZ

1.3.5 MI_BATCH_BUFFER_START

MI_BATCH_BUFFER_START			
Project:	All	**Length Bias:**	2
Engine:	**Render**		

The MI_BATCH_BUFFER_START command is used to initiate the execution of commands stored in a *batch buffer*. For restrictions on the location of batch buffers, see Batch Buffers in the Device Programming Interface chapter of *MI Functions*.

Programming Notes:

- Batch buffers referenced with physical addresses must not extend beyond the end of the starting physical page (can't span physical pages). However, a batch buffer initiated using a physical address can chain to another buffer in another physical page.
- 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.
- For virtual batch buffers, it is essential that the address location beyond the current page be populated inside the GTT. HW performs over-fetch of the command addresses and any over-fetch requires a valid TLB entry. A single extra page beyond the batch buffer is sufficient.
- Prior to sending batch buffer start command with clear command buffer enable set, software has to ensure pipe is flushed explicitly by sending MI_FLUSH or PIPE_CONTROL with CS Stall set..
- The dword following this command in the batch buffer should always be MI_NOOP.

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
	22:13	**Reserved** Project: All Format: MBZ
	12	**Batch Buffer Encrypted Memory Read Enable** Project: All Format: U1 The Command Streamer will request batch buffer data from serpent memory if this bit is enabled. If disabled then the batch buffer will be fetched from non-encrypted memory. Commands in the Table 3-7 Priviledged Commands are not allowed from Encryped Batch Buffers and will be turned into NOOP commands in the command streamer. Any write that is generated from the encrypted batch buffer will write encrypted data.
	11	**Clear Command Buffer Enable** Project: All Format: U1 The following batch buffer is to be executed from the Write Once protected memory area. The address of the batch buffer is an offset into the WOPCM area. This batch buffer needs to be pre-ceded by a MI_FLUSH command or PIPE_CONTROL with CS Stall set.
	10:9	**Reserved** Project: All Format: MBZ

<table>
<tr><th colspan="3">MI_BATCH_BUFFER_START</th></tr>
<tr><td></td><td>8</td><td>Buffer Security and Address Space Indicator
Project: All
Format: MI_BufferSecurityType
When this command is executed directly from a ring buffer, this field is used to specify the associated batch buffer as a secure or non-secure buffer. 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. 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.
<table><tr><th>Value Name</th><th>Description</th><th>Project</th></tr><tr><td>0h</td><td>MI_BUFFER_SECURE</td><td>This batch buffer is secure and will be accessed via the GGTT.</td><td>All</td></tr></table><table><tr><th>Programming Notes</th><th>Project</th></tr><tr><td>Notes</td><td>All</td></tr></table><table><tr><th>Errata Description</th><th>Project</th></tr><tr><td>#</td><td>Desc</td><td>All</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td>DWord Length
Default Value: 0h Excludes DWord (0,1)
Format: =n Total - Bias</td></tr>
<tr><td>1</td><td>31:2</td><td>Batch Buffer Start Address
Project: All
Address: GraphicsAddress[31:2]
Surface Type: BatchBuffer
This field specifies Bits 31:2 of the starting address of the batch buffer.</td></tr>
<tr><td></td><td>1:0</td><td>Reserved Project: All Format: MBZ</td></tr>
</table>

1.3.5.1 Command Access of Privileged Memory

Memory space mapped through the global GTT is considered “privileged” memory. Commands that have the capability of accessing both privileged and unprivileged (PPGTT space) memory will contain a bit that, if set, will attempt a “privileged” access through the GGTT rather than an unprivileged access through the context-local PPGTT.

“User mode” command buffers should not be able to access privileged memory under any circumstances. These command buffers will be issued by the kernel mode driver with the batch buffer’s **Buffer Security** Indicator set to “non-secure”. Commands in such a batch buffer are not allowed to access privileged memory. The commands in these buffers are supplied by the user mode driver and will not be validated by the kernel mode driver.

"Kernel mode" command buffers are allowed to access privileged memory. The batch buffers Buffer Security indicator is set to "secure" in this case. In some of the commands that access memory in a secure batch buffer, a bit is provided in the command to steer the access to Per process or Global virtual space. Secure batch buffers are executed from the global GTT.

Commands in ring buffers and commands in batch buffers that are marked as secure (by the kernel mode driver) are allowed to access both privileged and unprivileged memory and may choose on a command-by-command basis.

Table 1. GGTT and PPGTT Usage by Command

Command	Address	Allowed Access
MI_BATCH_BUFFER_START*	Command Address	Selectable
MI_DISPLAY_FLIP	Display Buffer Base	GGTT Only
MI_STORE_DATA_IMM*	Storage Address	Selectable
MI_STORE_DATA_INDEX**	Storage Offset	Selectable
MI_STORE_REGISTER_MEM*	Storage Address	Selectable
MI_SEMAPHORE_MBOX	Semaphore Address	Selectable
PIPE_CONTROL	STDW Address	Selectable

*Command has a GGTT/PPGTT selector added to it vs. previous Gen4 family products.

**Added bit allows offset to apply to global HW Status Page or PP HW Status Page found in context image.

1.3.5.2 Privileged Commands

A subset of the commands are privileged. These commands may be issued only from a secure batch buffer or directly from a ring. If one of these commands is parsed in a non-secure batch buffer, an error is flagged and the command is dropped. For commands that generates a write, the hardware will complete the transaction but the byte enables are turned off. Batch buffers from the User mode driver are passed directly to the kernel mode driver which does not validate them but issues them with the Security Indicator set to 'non-secure' to protect the system from an attack using these privileged commands.

Table 2. Privileged Commands

Privileged Command	Function in non-privileged batch buffers
MI_LOAD_REGISTER_IMM	Byte enables are turned off
MI_UPDATE_GTT	Byte enabled are turned off
MI_STORE_REGISTER_MEM	Command is translated and completed with byte enables turned off
MI_DISPLAY_FLIP	Command is ignored by the hardware

Command privilege applies the same way in Basic Scheduler mode. Parsing one of the commands in the table above from a non-secure batch buffer will flag an error and convert the command to a NOOP.

1.3.5.3 Privileged Commands [PreDevSNB]

A subset of the commands are privileged. These commands may be issued only from a secure batch buffer or directly from a ring. If one of these commands is parsed in a non-secure batch buffer, an error is flagged and the command is dropped. For commands that generates a write, the hardware will complete the transaction but the byte enables are turned off. Batch buffers from the User mode driver are passed directly to the kernel mode driver which does not validate them but issues them with the Security Indicator set to 'non-secure' to protect the system from an attack using these privileged commands.

Table 3. Privileged Commands

Privileged Command	Function in non-privileged batch buffers
MI_LOAD_REGISTER_IMM	Byte enables are turned off
MI_UPDATE_GTT	Byte enabled are turned off
MI_STORE_REGISTER_MEM	Command is translated and completed with byte enables turned off
MI_DISPLAY_FLIP	Command is ignored by the hardware

Command privilege applies the same way in Basic Scheduler mode. Parsing one of the commands in the table above from a non-secure batch buffer will flag an error and convert the command to a NOOP.

1.3.6 MI_DISPLAY_FLIP

MI_DISPLAY_FLIP			
Project:	All	**Length Bias:**	2
Engine:	Render		

The MI_DISPLAY_FLIP command is used to request a specific display plane to switch (flip) to display a new buffer. The buffer is specified with a starting address and pitch. The tiled attribute of the buffer start address is programmed as part of the packet. This command is specific to the render engine

The operation this command performs is also known as a "display flip request" operation – in that the flip operation itself will occur at some point in the future. This command specifies when the flip operation is to occur: either synchronously with vertical retrace to avoid tearing artifacts (possibly on a future frame), or asynchronously (as soon as possible) to minimize rendering stalls at the cost of tearing artifacts.

Programming Notes:

1. This command simply requests a display flip operation -- command execution then continues normally. There is no guarantee that the flip (even if asynchronous) will occur prior to subsequent commands being executed. (Note that completion of the MI_FLUSH command does not guarantee that outstanding flip operations have completed). The MI_WAIT_FOR_EVENT command can be used to provide this synchronization – by pausing command execution until a pending flip has actually completed. This synchronization can also be performed by use of the Display Flip Pending hardware status. See Display Flip Synchronization in the Device Programming Interface chapter of *MI Functions*.
2. After a display flip operation is requested, software is responsible for initiating any required synchronization with subsequent buffer clear or rendering operations. For multi-buffering (e.g., double buffering) operations, this will typically require updating SURFACE_STATE or the binding table to change the rendering (back) buffer. In addition, prior to any subsequent clear or rendering operations, software must typically ensure that the new rendering buffer is not actively being displayed. Again, the MI_WAIT_FOR_EVENT command or Display Flip Pending hardware status can be used to provide this synchronization. See Display Flip Synchronization in the Device Programming Interface chapter of *MI Functions.*
3. The display buffer command uses the X and Y offset for the tiled buffers from the Display Interface registers. Software is allowed to change the offset via the MMIO interface irrespective of the flip commands enqueued in the command stream. For tiled buffers, the display subsystem uses the X and Y offset in generation of the final request to memory. The offset is always updated on the next vblank for both Synchronous and Asynch Flips. It is not necessary to have a flip enqueued to update the X and Y offset
4. The display buffer command uses the linear dword offset for the linear buffers from the Display Interface registers. Software is allowed to change the offset via the MMIO interface irrespective of the flip commands enqueued in the command stream. For linear buffers, the display subsystem uses the dword offset in generation of the final request to memory.
   - For synchronous flips the offset is updated on the next vblank. It is not necessary to have a sync flip enqueued to update the dword offset.
   - Linear memory does not support asynchronous flips
5. DWord 3 (panel fitter flip) must not be sent with asynchronous flips. It is only allowed to be sent with synchronous flips.

MI_DISPLAY_FLIP

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 14h MI_DISPLAY_FLIP Format: OpCode
	22	**Async Flip Indicator** Project: All Format: Enable This bit should always be set if DW2 [1:0] == '01' (async flip). This field is required due to HW limitations. This bit is used by the render pipe while DW2 is used by the display hardware.
	18:8	**Reserved** Project: Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:16	**Reserved** Project: All Format: MBZ
	15:6	**Display Buffer Pitch** Project: All Default Value: 0h DefaultVaueDesc Format: U10 *For Synchronous Flips only*, this field specifies the 64-byte aligned pitch of the new display buffer. For Asynchronous Flips, this parameter is programmed so that all the flips in a flip chain should maintain the same pitch as programmed with the last synchronous flip or direct thru mmio.
	5:1	**Reserved** Project: All Format: MBZ

Project:	All	**Length Bias:**	2
Engine:	Render		

The MI_DISPLAY_FLIP command is used to request a specific display plane to switch (flip) to display a new buffer. The buffer is specified with a starting address and pitch. The tiled attribute of the buffer start address is programmed as part of the packet. This command is specific to the render engine

The operation this command performs is also known as a "display flip request" operation – in that the flip operation itself will occur at some point in the future. This command specifies when the flip operation is to occur: either synchronously with vertical retrace to avoid tearing artifacts (possibly on a future frame), or asynchronously (as soon as possible) to minimize rendering stalls at the cost of tearing artifacts.

Programming Notes:

6. This command simply requests a display flip operation -- command execution then continues normally. There is no guarantee that the flip (even if asynchronous) will occur prior to subsequent commands being executed. (Note that completion of the MI_FLUSH command does not guarantee that outstanding flip operations have completed). The MI_WAIT_FOR_EVENT command can be used to provide this synchronization – by pausing command execution until a pending flip has actually completed. This synchronization can also be performed by use of the Display Flip Pending hardware status. See Display Flip Synchronization in the Device Programming Interface chapter of *MI Functions*.
7. After a display flip operation is requested, software is responsible for initiating any required synchronization with subsequent buffer clear or rendering operations. For multi-buffering (e.g., double buffering) operations, this will typically require updating SURFACE_STATE or the binding table to change the rendering (back) buffer. In addition, prior to any subsequent clear or rendering operations, software must typically ensure that the new rendering buffer is not actively being displayed. Again, the MI_WAIT_FOR_EVENT command or Display Flip Pending hardware status can be used to provide this synchronization. See Display Flip Synchronization in the Device Programming Interface chapter of *MI Functions.*
8. The display buffer command uses the X and Y offset for the tiled buffers from the Display Interface registers. Software is allowed to change the offset via the MMIO interface irrespective of the flip commands enqueued in the command stream. For tiled buffers, the display subsystem uses the X and Y offset in generation of the final request to memory. The offset is always updated on the next vblank for both Synchronous and Asynch Flips. It is not necessary to have a flip enqueued to update the X and Y offset
9. The display buffer command uses the linear dword offset for the linear buffers from the Display Interface registers. Software is allowed to change the offset via the MMIO interface irrespective of the flip commands enqueued in the command stream. For linear buffers, the display subsystem uses the dword offset in generation of the final request to memory.
    - For synchronous flips the offset is updated on the next vblank. It is not necessary to have a sync flip enqueued to update the dword offset.
    - Linear memory does not support asynchronous flips
10. DWord 3 (panel fitter flip) must not be sent with asynchronous flips. It is only allowed to be sent with synchronous flips.

<table>
<tr><td>2</td><td>31:12</td><td>Display Buffer Base Address
Project: All
Address: GraphicsAddress[31:12]
This field specifies Bits 31:12 of the Graphics Address of the new display buffer. (Refer to the Display Address Start Address Register description in the Display Registers chapter).

Programming Notes
• The Display buffer must reside completely in Main Memory
• This address is always translated via the global (rather than per-process) GTT</td></tr>
<tr><td></td><td>1:0</td><td>Flip Type
Project: All
Default Value: 00h Synchronous flip
This field specifies whether the flip operation should be performed asynchronously to vertical retrace.

<table><tr><th>Value Name</th><th>Description</th><th>Project</th></tr><tr><td>00h</td><td>Sync Flip</td><td>The flip will occur during the vertical blanking interval – thus avoiding any tearing artifacts.</td><td>All</td></tr><tr><td>01h</td><td>Async Flip</td><td>The flip will occur “as soon as possible” – and may exhibit tearing artifacts</td><td>All</td></tr><tr><td>1Xh</td><td>Reserved</td><td></td><td>All</td></tr></table>
Programming Notes
• The Display Buffer Pitch and Tile parameter fields cannot be changed for asynchronous flips (i.e., the new buffer must have the same pitch/tile format as the previous buffer).
• Supported on X-Tiled Frame buffers only.
• For Asynch Flips the Buffers used must be 32KB aligned.
• Supported on Display Planes A and B and C only</td></tr>
<tr><td>3</td><td>31</td><td>Enable Panel Fitter Project: All Format: Enable
Enables the panel fitter on the pipe attached to the plane selected for this flip.</td></tr>
<tr><td></td><td>30:28</td><td>Reserved Project: All Format: MBZ</td></tr>
</table>

	27:16	**Pipe Horizontal Source Image Size** Project: All Format: U32 This 12-bit field specifies Horizontal source image size up to 4096. This determines the size of the image created by the display planes sent to the blender. The value programmed should be the source image size minus one. This field obeys all the rules of the Horizontal Source Image Size registers. The pipe affected will be the pipe attached to the plane selected for this flip.
	15:12	**Reserved** Project: All Format: MBZ
	11:0	**Pipe Vertical Source Image ReSize** Project: All Format: U32 This 12-bit field specifies the new vertical source image size up to 4096 lines. This determines the size of the image created by the display planes sent to the blender. The value programmed should be the source image size minus one. This field obeys all the rules of the Vertical Source Image Size registers. The pipe affected will be the pipe attached to the plane selected for this flip.

1.3.7 MI_FLUSH

MI_FLUSH			
Project:	All	**Length Bias:**	1
Engine:	**Render**		

The MI_FLUSH command is used to perform an internal "flush" operation. The parser pauses on an internal flush until all drawing engines have completed any pending operations and the read caches are invalidated including the texture cache accessed via the Sampler or the data port. In addition, this command can also be used to:

1. Flush any dirty data in the Render Cache to memory. This is done by default, however this can be inhibited.
2. Invalidate the state and command cache.

Usage note: After this command is completed and followed by a Store DWord-type command, CPU access to graphics memory will be coherent (assuming the Render Cache flush is not inhibited). This command is specific to the render engine. Other engines use MI_FLUSH_DW

[DevSNB]: This command is considered deprecated and will be removed completely in future projects. If it must still be used, enable bit 12 in the MI_MODE (0x209C) register

Note that if no post-sync operation is enabled for Flush completion, a register write to DE scratch space will be generated by command streamer. Scratch space description is given in DE Bspecs.

MI_FLUSH

DWord Bit		Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 04h MI_FLUSH Format: OpCode
	22:7	**Reserved** Project: All Format: MBZ
	6	**Protected memory Enable** Project: All Format: Enable 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.
	5	**Indirect State Pointers Disable** Project: All Format: Disable At the completion of the flush, the indirect state pointers in the hardware will be considered as invalid ie the indirect pointers will not be restored for the context.
	4	**Generic Media State Clear** Project: All Format: Disable If set, all generic media state context information will not be included with the next context save, assuming no new state is initiated after the flush. If clear, the generic media state context save state will not be affected. An MI_FLUSH with this bit set should be issued once all the Media Objects that will be processed by a given persistent root thread have been issued or when an MI_SET_CONTEXT switching from a generic media context to a 3D context completes. When using MI_SET_CONTEXT, once state is programmed, it will be saved and restarted as part of any context each time that context is saved/restored until an MI_FLUSH with this bit set is issued in that context.
	3	**Global Snapshot Count Reset** Project: All Format: Boolean

Programming Notes	Project
TIMESTAMP are *not* reset by MI_FLUSH with this bit set. TIMESTAMP and PS_DEPTH_COUNT can be reset by writing 0 to them	All

Value Name	Description	Project
0h Don't Reset	Do not reset the snapshot counts or Statistics Counters.	All
1h Reset	Reset the snapshot count in Gen4 for all the units and reset the Statistics Counters except as noted above.	All

2	**Render Cache Flush Inhibit** Project: All Format: Boolean If set, the Render Cache is not flushed as part of the processing of this command.	

Value Name	Description	Project
0h Flush	Flush the Render Cache	All
1h Don't Flush	Do not flush the Render Cache	All

MI_FLUSH		
	1	**State/Instruction Cache Invalidate** Project: All Format: Boolean If set, Invalidates the State and Instruction Cache **Value Name** / **Description** / **Project** 0h — Don't Invalidate — Leave State/Instruction Cache unaffected — All 1h — Invalidate — Invalidate State/Instruction Cache — All
	0	**Reserved** Project: All Format: MBZ

1.3.8 MI_LOAD_REGISTER_IMM

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

The MI_LOAD_REGISTER_IMM command requests a write of up to a DWord constant supplied in the command to the specified Register Offset (i.e., offset into Memory-Mapped Register Range).

Programming Notes:

- A stalling flush must be sent down pipeline before issuing this command
- 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 BB 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 insecure then the command stream converts this command to a NOOP. Note that the corresponding ring buffer must allow a register update for this command to execute.
- **To ensure this command gets executed before upcoming commands in the ring, either a stalling pipeControl should be sent after this command, or MMIO 0x20C0 bit 7 should be set to 1.**

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** Format: Enable[4] Bit 8 corresponds to Data DWord [7:0] Range Must specify a valid register write operation This field has only 2 options. If [11:8] is '1111', then the register write will not occur. Any other value and the register write will be fully written.

MI_LOAD_REGISTER_IMM		
	7:0	**DWord Length** Default Value: 1h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:2	**Register Offset** Format: U30 Address: MmioAddress[31:2] This field specifies bits [31:2] of the offset into the Memory Mapped Register Range (i.e., this field specifies a DWord offset).
	1:0	**Reserved** Project: All Format: MBZ
2	31:0	**Data DWord** Mask: Bytes Write Disables Format: U32 This field specifies the DWord value to be written to the targeted location.

1.3.9 MI_NOOP

MI_NOOP			
Project:	All	**Length Bias:**	1
Engine:	**Render**		

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

Performance Note: On [Pre-DevSNB, Pre-DEVILK] The process time to execute a NOP command is min of 6 clock cycles. On [DEVILK] The NOP process time is reduced to 1 clock. One example usage of the improved NOP throughput is for some multi-pass media application whereas some unwanted media object commands are replaced by MI_NOOP without repacking the commands in a batch buffer.

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

MI_NOOP

	22	**Identification Number Register Write Enable** Project: All Format: Enable This field enables the value in the Identification Number field to be written into the MI NOPID register. If disabled, that register is unmodified – making this command an effective "no operation" function.

Value Name	Description	Project
0h	Disable	Do not write the NOP_ID register.	All
1h	Enable	Write the NOP_ID register.	All

	31:0	**Identification Number** Project: All Format: U22 This field contains a 22-bit number which can be written to the MI NOPID register.

1.3.10 Surface Probing

These commands are only valid when the “Surface Fault Enable” bit is set in the GFX_MODE register

1.3.10.1 MI_PROBE

MI_PROBE			
Project:	All	**Length Bias:**	2
Engine:	**Render**		

The probe command is inserted into a ring or batch buffer in order to validate the base address(es) of a surface(s) required by subsequent commands. When parsed, the probe command will do a “test” access of the surface base address to see if it is valid. The probe will also be written to the specified slot of a memory-based probe list such that it can be re-validated if the current context is switched out and then switched back in. If the test access encounters an invalid page table entry, it said to “fault”. Faulting probes will trigger the current context to be switched.

A probe command containing multiple probes will process all of them regardless of which ones fault. If any probe faulted and the pipeline is busy, the next command (unless it is a probe or unprobe command) will stall until the pipeline drains. Once the pipeline is empty, the pending probes will be written to the probe list with the faulted probes ndicated and a context switch will occur.

Note that surfaces accessed through the global GTT need not be validated. It is assumed that Global GTT pages will not be invalidated while a context is switched out. Probe and unprobe are not privileged commands. The probe command can be used to insert only 512 probes in one command. Note that the total number of probes allowed in the system is 1024.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 25h MI_PROBE Format: OpCode
	22:10	**Reserved** Project: All Format: MBZ
	9:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1..n	31:12	**Surface Page Base Address** Project: All Address: PerProcessGraphicsVirtualAddress[31:12] Surface Type: U32 Range 0..2^32-1 The Per Process Address to validate.
	11:10	**Reserved** Project: All Format: MBZ

MI_PROBE		
	9:0	**Slot Number** Project: All Format: ProbeSlotIndex Range [0,1023] The index into the probe list where this probe will be stored.

1.3.10.2 MI_UNPROBE

MI_UNPROBE			
Project:	All	**Length Bias:**	1
Engine:	**Render**		

There are 2 ways to remove probes. SW may issue a new probe to the same slot as an existing probe (presumably with a new surface base address), and the old probe will be replaced with the new, effectively deleting the old probe. If it has no new probe to place in the slot, SW may issue the unprobe command to remove probes by invaliding probe slots.

The unprobe command is used to remove probes from the probe list. No **Surface Address** is provided; the specified slot is simply marked invalid. The Unprobe command does not affect the probe list in memory; it only clears probe **Slot Valid** bits in the Probe List Slot Valid Registers (see *Memory Interface Registers*).

DWord Bit		Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 06h MI_UNPROBE Format: OpCode
	22:10	**Reserved** Project: All Format: MBZ
	9:0	**Slot Number** Project: All Format: ProbeSlotIndex Range [0,1023] The probe list index of the probe to be removed.

1.3.11 MI_REPORT_HEAD

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

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

The location written is relative to the address programmed in the Hardware Status Page Address Register.

Programming Notes:

- This command must not be executed from a Batch Buffer (Refer to the description of the HSW_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

1.3.12 MI_SEMAPHORE_MBOX

MI_SEMAPHORE_MBOX			
Project:	All	**Length Bias:**	2
Engine:	**Render**		

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.
	19	**Reserved** Project: All Format: MBZ

MI_SEMAPHORE_MBOX		
	18	**Compare Register** Project: All 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	**Register Select** Project: All Format: Register Select If compare register is set in bit[18], this filed indicate which register will be used. 0: VCS register (RVSYNC) 1: BCS regiser (RBSYNC)
	16:8	**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 is 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

1.3.13 MI_SET_CONTEXT

MI_SET_CONTEXT			
Project:	All	**Length Bias:**	2
Engine:	**Render**		

The MI_SET_CONTEXT command is used to specify the *logical* context associated with the hardware context. A logical context is an area in memory used to store hardware context information, and the context is referenced via a 2KB-aligned pointer. If the (new) logical context is different (i.e., at a different memory address), the device will proceed to save the current HW context values to the current logical context address, and then restore (load) the new logical context by reading the context from the new address and loading it into the hardware context state. If the logical context address specified in this command matches the current logical context address, this command is effectively treated as a NOP.

This command also includes some controls over the context save/restore process. It is specific to the render engine

- The **Force Restore** bit can be used to refresh the on-chip device state from the same memory address if the indirect state buffers have been modified.
- The **Restore Inhibit** bit can be used to prevent the new context from being loaded at all. This **must** be used to prevent an uninitialized context from being loaded. Once software has initialized a context (by setting all state variables to initial values via commands), the context can then be stored and restored normally.
- This command needs to be always followed by a single MI_NOOP instruction to workaround a Gen4 silicon issue.
- When switching from a generic media context to a 3D context, the generic media state must be cleared via the *Generic Media State Clear* bit 16 in PIPE_CONTROL (or bit 4 in MI_FLUSH) before saving 3D context.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 18h MI_SET_CONTEXT Format: OpCode
	22:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2

<table>
<tr><td rowspan="8">1</td><td>31:12</td><td>Logical Context Address
Project: All
Address: GraphicsAddress[31:12]
Surface Type: Logical Context
This field contains the 4KB-aligned physical address of the Logical Context that is to be loaded into the hardware context. If this address is equal to the CCID register associated with the current ring, no load will occur. Prior to loading this new context, the device will save the existing context as required. After the context switch operation completes, this address will be loaded into the associated CCID register.

<table>
<tr><th colspan="2">[DevSNB A]</th></tr>
<tr><th>Description Ring</th><th>Command</th></tr>
<tr><td>Switch to default context</td><td>MI_SET_CONTEXT save old_ctx, restore default ctx</td></tr>
<tr><td>Nuke default context</td><td>MI_LOAD_REGISTER_IMM address 0x2180, data = 0x0</td></tr>
<tr><td>Wait for nuking to complete</td><td>PIPE_CONTROL with CS stall (bit20 in DW1) bit set (PIPE_CONTROL restrictions apply)</td></tr>
<tr><td>Switch to new context</td><td>MI_SET_CONTEXT restore new ctx</td></tr>
</table></td></tr>
<tr><td>11:10</td><td>Reserved Project: All Format: MBZ</td></tr>
<tr><td>9</td><td>Reserved Project: Format: MBZ</td></tr>
<tr><td>8</td><td>Reserved, Must be 1 Project: All Format: Must Be One</td></tr>
<tr><td>7:4</td><td>Reserved Project: All Format: MBZ</td></tr>
<tr><td>1</td><td>Force Restore Project: All Format: U32
When switching to this logical context a comparison between Logical Context Address and the contests of the CCID register is performed. Normally, matching addresses prevent a context restore from occurring; however, when this bit is set a context restore is forced to occur. This bit cannot be set with Restore Inhibit.
Note: This bit is not saved in the associated CCID register. It only affects the processing of this command.</td></tr>
<tr><td>0</td><td>Restore Inhibit Project: All Format: U32
If set, the restore of the HW context from the logical context specified by Logical Context Address is inhibited (i.e., the existing HW context values are maintained). This bit must be used to prevent the loading of an uninitialized logical context. If clear, the context switch proceeds normally. This bit cannot be set with Force Restore.
Note: This bit is not saved in the associated CCID register. It only affects the processing of this command.</td></tr>
</table>

1.3.14 MI_STORE_DATA_IMM

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

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

Programming Notes:
This command 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.
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 This bit will be ignored and treated as if clear when executing from a non-privileged batch buffer. It is allowed for this bit to be clear when executing this command from a privileged (secure) batch buffer. This bit *must* be '1' if the **Per Process GTT Enable** bit is clear.
	21:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 2h Excludes DWord (0,1) = 2 for DWord, 3 for QWord Format: =n Total Length - 2
1	31:0	**Reserved** Project: All Format: MBZ

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

MI_STORE_DATA_IMM		
2	31:2	**Address** Project: All Address: GraphicsAddress[31:2] Surface Type: U32(2) This field specifies Bits 31:2 of the Address where the DWord will be stored. As the store address must be DWord-aligned, Bits 1:0 of that address MBZ. This address must be 8B aligned for a store “QW” command.
	1:0	**Reserved** Project: All Format: MBZ
3	31:0	**Data DWord 0** Project: All Format: U32 This field specifies the DWord value to be written to the targeted location. For a QWord write this DWord is the lower DWord of the QWord to be reported (DW 0).
4	31:0	**Data DWord 1** Project: All Format: U32 This field specifies the upper DWord value to be written to the targeted QWord location (DW 1).

1.3.15 MI_STORE_DATA_INDEX

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

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

MI_STORE_DATA_INDEX		
	22	**Reserved** Project: CTG+ Format: Setting this bit will cause this command to offset in the Surface Probe List instead of the hardware status page. This is intended to be used internally only (it is UNDEFINED to set this bit in a command in a ring or batch buffer.)
	21	**Use Per-Process Hardware Status Page** Project: All If this bit is set, this command will index into the per-process hardware status page at offset 28K from the LRCA. If clear, the Global Hardware Status Page will be indexed. This bit will be ignored and treated as set if this command is executed from within a non-secure batch buffer, This
	20:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 1h Excludes DWord (0,1) = 1 for DWord, 2 for QWord Format: =n Total Length - 2
1	31:12	**Reserved** Project: All Format: MBZ
	11:2	**Offset** Project: All Format: U10 zero-based DWord offset into the HW status page. Address: HardwareStatusPageOffset[11:2] 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. This address must be 8B aligned for a store “QW” command.
	1:0	**Reserved** Project: All Format: MBZ
2	31:0	**Data DWord 0** Project: All Format: U32 This field specifies the DWord value to be written to the targeted location. For a QWord write this DWord is the lower DWord of the QWord to be reported (DW 0).
3	31:0	**Data DWord 1** Project: All Format: U32 This field specifies the upper DWord value to be written to the targeted QWord location (DW 1).

1.3.16 MI_STORE_REGISTER_MEM

MI_STORE_REGISTER_MEM			
Project:	All	**Length Bias:**	2
Engine:	**Render**		

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

The command temporarily halts command execution.

The memory address for the write is snooped on the host bus.

This command should not be used 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.

This command will cause undefined data to be written to memory if given register addresses for the PGTBL_CTL_0 or FENCE registers

SNB-A0: To avoid deadlock scenarios, this command cannot be executed if there are additional posted writes (i.e. LRI, semaphore update) being sent to the same command streamer.

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 will be ignored and treated as if clear when executing from a non-privileged batch buffer. It is allowed for this bit to be clear when executing this command from a privileged (secure) batch buffer. This bit *must* be '1' if the **Per Process GTT Enable** bit is clear.
	21:8	**Reserved** Project: All Format: MBZ

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

MI_STORE_REGISTER_MEM

	7:0	**DWord Length** Default Value: 1h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:26	**Reserved** Project: All Format: MBZ
	25:2	**Register Address** Project: All Address: MMIO Address[25:2] Surface Type: MMIO Register This field specifies Bits 25:2 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: All 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

1.3.17 MI_SUSPEND_FLUSH

MI_SUSPEND_FLUSH			
Project:	All	**Length Bias:**	1
Engine:	**Render**		

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

1.3.17.1 Description of Dedicated Performance Counters [A0-A28]

Cntr #	Event	Description
A0	Aggregated Core Array Active	The sum of all cycles on all cores spent actively executing instructions.
A1	Aggregated Core Array Stalled	The sum of all cycles on all cores spent stalled. (at least one thread loaded but the entire core is stalled for any reason)
A2	Vertex Shader Active Time	Total time in clocks the vertex shader spent active on all cores.
A3	Vertex Shader Stall Time	Total time in clocks the vertex shader spent stalled on all cores. This metric must be bucketed by stall type (“other” is ok – but must have buckets for things that are architecturally interesting)
A4	Vertex Shader Stall Time – Core Stall	Total time in clocks the vertex shader spent stalled on all cores – and the entire core was stalled as well. This metric must be bucketed by stall type (“other” is ok – but must have buckets for things that are architecturally interesting)
A5	Vertex Shader ready but not running Time	Total time in clocks the vertex shader spent ready to run but not running on all cores.
A6	Geometry Shader Active Time	Total time in clocks the geometry shader spent active on all cores.
A7	Geometry Shader Stall Time	Total time in clocks the geometry shader spent stalled on all cores. This metric must be bucketed by stall type (“other” is ok – but must have buckets for things that are architecturally interesting)
A8	Geometry Shader Stall Time – Core Stall	Total time in clocks the geometry shader spent stalled on all cores – and the entire core was stalled as well. This metric must be bucketed

Cntr #	Event	Description
		by stall type (“other” is ok – but must have buckets for things that are architecturally interesting)
A9	# GS threads loaded	Number of GS threads loaded at any given time in the EUs.
A10	Geometry Shader ready but not running Time	Total time in clocks the geometry shader spent ready to run but not running on all cores.
A11	Pixel Shader Active Time	Total time in clocks the pixel shader spent active on all cores.
A12	Pixel Shader Stall Time	Total time in clocks the Pixel shader spent stalled on all cores. This metric must be bucketed by stall type (“other” is ok – but must have buckets for things that are architecturally interesting)
A13	Pixel Shader Stall Time – Core Stall	Total time in clocks the pixel shader spent stalled on all cores – and the entire core was stalled as well. This metric must be bucketed by stall type (“other” is ok – but must have buckets for things that are architecturally interesting)
A14	# PS threads loaded	Number of PS threads loaded at any given time in the EUs.
A15	Pixel Shader ready but not running Time	Total time in clocks the Pixel shader spent ready to run but not running on all cores.
A16	Early Z Test Pixels Passing	Number of pixels/samples passing early Z test (i.e. before PS dispatch)
A17	Early Z Test Pixels Failing	Number of pixels/samples failing early Z test (i.e. before PS dispatch)
A18	Early Stencil Test Pixels Passing	Number of pixels/samples passing early stencil test (i.e. before PS dispatch)
A19	Early Stencil Test Pixels Failing	Number of pixels/samples failing early stencil test (i.e. before PS dispatch)
A20	Pixel Kill Count	Number of pixels/samples killed in the pixel shader. (How about chroma key?)

Cntr #	Event	Description
A21	Alpha Test Pixels Failed	Number of pixels/samples that fail alpha-test. Alpha to coverage may have some challenges in per-pixel invocation.
A22	Post PS Stencil Pixels Failed	Number of pixels/samples fail stencil test in the backend.
A23	Post PS Z buffer Pixels Failed	Number of pixels/samples fail Z test in the backend.
A24	Pixels/samples Written in the frame buffer	MRT case will report multiple of those.
A25	*GPU Busy*	CSunit indicating that ring is idle.
A26	*CL active and not stalled*	Clipper Fixed Function is active but not stalled
A27	*SF active and stalled*	SF Fixed Function is active but not stalled

1.3.18 MI_UPDATE_GTT

MI_UPDATE_GTT			
Project:	All	**Length Bias:**	2
Engine:	Render		

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.

PPGTT updates cannot be done via MI_UPDATE_GTT, gfx driver will have to use storeDW for PPGTT inline updates.

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
	21:8	**Reserved** Project: All Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2

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		
1	31:12	**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.
	11:0	**Reserved** Project: All Format: MBZ
2..n	31:0	**Entry Data** Project: All Format: Table Entry This Dword becomes the new page table entry. See PPGTT/Global GTT Table Entries (PTEs) in Memory Interface Registers.

1.3.19 MI_USER_INTERRUPT

MI_USER_INTERRUPT			
Project:	All	**Length Bias:**	1
Engine:	**Render**		

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** Project: All Format: MBZ

1.3.20 MI_WAIT_FOR_EVENT

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

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
	28:23	**MI Command Opcode** Default Value: 03h MI_WAIT_FOR_EVENT Format: OpCode
	22:19	**Reserved** Project: All Format: MBZ
	18	**Reserved** Project: BW Format: MBZ
	18	**Display Pipe B Start of V Blank Wait Enable** Project: All Format: Enable This field enables a wait until the start of next Display Pipe B “Vertical Blank” event occurs. This event is defined as the start of the next Display B Vertical blank period. Note that this can cause a wait for up to a frame. See Start of Vertical Blank Event in the Device Programming Interface chapter of *MI Functions*. **Errata Description** — **Project** BWT013 — MBZ — BW
	17	**Reserved** Project: BW Format: MBZ

MI_WAIT_FOR_EVENT

	17	**Display Pipe A Start of V Blank Wait Enable** Project: CL+ Format: Enable This field enables a wait until the start of next Display Pipe A "Vertical Blank" event occurs. This event is defined as the start of the next Display A Vertical blank period. Note that this can cause a wait for up to a frame. See Start of Vertical Blank Event in the Device Programming Interface chapter of *MI Functions*.

Programming Notes	Project
Notes	All

Errata De	scription	Project
BWT013	MBZ	BW

	16	**Overlay Flip Pending Wait Enable** Project: BW,CL Format: Enable This field enables a wait for the duration of an Overlay "Flip Pending" condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new overlay address has been loaded into the corresponding overlay registers). See Overlay Flip Pending Condition in the Device Programming Interface chapter of *MI Functions*.
	16	**Display Sprite B Flip Pending Wait Enable** Project: CTG+ Format: Enable This field enables a wait for the duration of a Display Sprite B "Flip Pending" condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of *MI Functions*.
	15	**Reserved** Project: All Format: MBZ
	14	**Display Pipe B H Blank Wait Enable** Project: All Format: Enable This field enables a wait until the start of next Display Pipe B "Horizontal Blank" event occurs. This event is defined as the start of the next Display B Horizontal blank period. Note that this can cause a wait for up to a line. See Horizontal Blank Event in the Device Programming Interface chapter of *MI Functions*.
	13	**Display Pipe A H Blank Wait Enable** Project: All Format: Enable This field enables a wait until the start of next Display Pipe A "Horizontal Blank" event occurs. This event is defined as the start of the next Display A Horizontal blank period. Note that this can cause a wait for up to a line. See Horizontal Blank Event in the Device Programming Interface chapter of *MI Functions*.

12:9 **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	Enabled	Condition Code select enabled; selects one of 5 codes, 0 – 4	All
6h-15h	Reserved		All

Programming Notes	Project
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.	All

8 **Display Plane C Flip Pending Wait Enable** Project: BW,CL Format: Enable

This field enables a wait for the duration of a Display Plane C "Flip Pending" condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of *MI Functions*.

8 **Display Sprite A Flip Pending Wait Enable** Project: CTG+ Format: Enable

This field enables a wait for the duration of a Display Sprite A "Flip Pending" condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of *MI Functions*.

7 **Display Pipe B Vertical Blank Wait Enable** Project: All Format: Enable

This field enables a wait until the next Display Pipe B "Vertical Blank" event occurs. This event is defined as the start of the next Display Pipe B vertical blank period. Note that this can cause a wait for up to an entire refresh period. See Vertical Blank Event (See *Programming Interface*).

Programming Notes	Project
Prior to using the MI_WAIT_FOR_EVENT command to wait on Display Pipe A/B VBlank events, the corresponding Vertical Blank Interrupt Enable (bit 17) of the corresponding PIPEASTAT (70024h) or PIPEBSTAT (71024h) register must be set. Note that this does not require an actual VBlank interrupt to be enabled.	All

6 **Display Plane B Flip Pending Wait Enable** Project: All Format: Enable

This field enables a wait for the duration of a Display Plane B "Flip Pending" condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of *MI Functions*.

MI_WAIT_FOR_EVENT

	5	**Display Pipe B Scan Line Window Wait Enable** Project: All Format: Enable This field enables a wait while a Display B "In Scan Line Window" condition exists. This condition is defined as the period of time the Display B refresh is inside the scan line window as specified by a previous MI_LOAD_SCAN_LINES_INCL or MI_LOAD_SCAN_LINES_EXCL command. If the Display B refresh is outside this window, or a window has not been specified, the parser proceeds, treating this command as a no-op. If the Display B refresh is currently inside this window, the parser will wait until the refresh exits the window. See Scan Line Window Condition in the Device Programming Interface chapter of *MI Functions*.
	4	**Frame Buffer Compression Idle Wait Enable** Project: All Format: Enable This field enables a wait while the Frame Buffer compressor is busy. The ring that this command got executed from is removed from arbitration for the wait period and is inserted into arbitration as soon as the frame buffer compressor is idle.
	3	**Display Pipe A Vertical Blank Wait Enable** Project: All Format: Enable This field enables a wait until the next Display Pipe A "Vertical Blank" event occurs. This event is defined as the start of the next Display A vertical blank period. Note that this can cause a wait for up to an entire refresh period. See Vertical Blank Event in the Device Programming Interface chapter of *MI Functions*.

Programming Notes	Project
Prior to using the MI_WAIT_FOR_EVENT command to wait on Display Pipe A/B VBlank events, the corresponding Vertical Blank Interrupt Enable (bit 17) of the corresponding PIPEASTAT (70024h) or PIPEBSTAT (71024h) register must be set. Note that this does not require an actual VBlank interrupt to be enabled.	All

	2	**Display Plane A Flip Pending Wait Enable** Project: All Format: Enable This field enables a wait for the duration of a Display Plane A "Flip Pending" condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of *MI Functions*.
	1	**Display Pipe A Scan Line Window Wait Enable** Project: All Format: Enable This field enables a wait while a Display Pipe A "In Scan Line Window" condition exists. This condition is defined as the period of time the Display A refresh is inside the scan line window as specified by a previous MI_INCLUSIVE_SCAN_WINDOW or MI_EXCLUSIVE_SCAN_WINDOW command. If the Display A refresh is outside this window, or a window has not been specified, the parser proceeds, treating this command as a no-op. If the Display A refresh is currently inside this window, the parser will wait until the refresh exits the window. See Scan Line Window Condition in the Device Programming Interface chapter of *MI Functions*.
	0	**Reserved** Project: All Format: MBZ