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

Volume 1 Part 3: Graphics Core – Memory Interface and Commands for the Render Engine (SandyBridge)

For the 2011 Intel Core Processor Family

May 2011

Revision 1.0

NOTICE:

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

Doc Ref #: IHD-OS-V1 Pt3 – 05 11

Contents

1. Render Engine Command Streamer

[DevSNB-D2] On hard boot, the command streamer must be programmed as follows to work-around a known issue that affects power management. This is expected to be done in BIOS

1) Disable CSunit level clock gating

2) Reset Render pipe

1.1 Registers in Render Engine

1.1.1 Introduction

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

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.

1.1.1.1 ARB_MODE – Arbiter Mode Control Register [DevSNB]

ARB_MODE – Arbiter Mode Control Register	
Register Type:	MMIO_CS
Address Offset:	4030h
Project:	DevSNB+
Default Value:	00000000h
Access:	R/W
Size (in bits):	32
Trusted Type:	1

Bit	Description
31:16	**Mask bits** Project: DevSNB+ Format: U16 Mask bits act as write enables for the bits in the lower bits of this register
15:9	**Reserved** Project: All Format: MBZ
8	**Reserved**
7:6	**Reserved** Project: All Format:

ARB_MODE – Arbiter Mode Control Register	
5:4	**Address Swizzling for Tiled-Surfaces** Project: All Format: U1 This register location is updated via GFX Driver prior to enabling DRAM accesses. Driver needs to obtain the need for memory address swizzling via DRAM configuration registers and set the following bits (in Display Engine and Render/Media access streams)

Value	Name	Description	Project
00	No address Swizzling	No address Swizzling	DevSNB+
01	Address bit[6] needs to be swizzled for tiled surfaces	Address bit[6] needs to be swizzled for tiled surfaces	DevSNB+
10		Reserved	DevSNB+
11		Reserved	DevSNB+

3:0	**Reserved**

1.1.1.2 ARB_WR_GAC_GAM3 – GAC_GAM WR Arbitration Register 3

ARB_WR_GAC_GAM3	
Register Type:	MMIO_CS
Address Offset:	43FCh
Pwrject:	DevSNB+
Default Value:	00000000h
Access:	R/W
Size (in bits):	32
Trusted Type:	1

Bit	Description
31:28	**Reserved**
27	Priority for entry 7
26:24	Goto field for entry 7 when request vector is 11.
23:21	Goto field for entry 7 when request vector is 10.
20:18	Goto field for entry 7 when request vector is 01.
17:15	Goto field for entry 7 when request vector is 00.
14:13	**Reserved**
12	Priority for entry 6
11:9	Goto field for entry 6 when request vector is 11.
8:6	Goto field for entry 6 when request vector is 10.
5:3	Goto field for entry 6 when request vector is 01.
2:0	Goto field for entry 6 when request vector is 00.

1.1.2 Virtual Memory Control

1.1.2.1 HWS_PGA — Hardware Status Page Address Register

HWS_PGA — Hardware Status Page Address Register	
Register Type:	MMIO_CS
Address Offset:	4080h
Project:	All
Default Value:	UUUU0000h
Access:	R/W
Size (in bits):	32
Trusted Type:	1

This register is used to program the 4 KB-aligned System Memory address of the Hardware Status Page used to report hardware status into (typically cacheable) System Memory. **[DevSNB]** This address in this register is translated using the Global GTT in memory. The mapping type of the GTT entry determines the snoop nature of the transaction to memory.

Bit	Description
31:12	**Address** Project: All Security: None Address: GraphicsAddress[31:12] This field is used by SW to specify Bits 31:12 of the 4 KB-aligned System Memory address of the 4 KB page known as the “Hardware Status Page”. The Global GTT is used to map this page from the graphics virtual address to physical address **Programming Notes** If the **Per-Process Virtual Address Space** is set, HW requires that the status page is programmed to allow for the context switch status to be reported
11:0	**Reserved** Project: All Format: MBZ

The following table defines the layout of the Hardware Status Page:

DWord Offset	Description
0	**Interrupt Status Register Storage:** The content of the ISR register is written to this location whenever an “unmasked” bit of the ISR (as determined by the HWSTAM register) changes state.
3:1	**Reserved.** Must not be used.
4	**Ring Head Pointer Storage:** The contents of the Ring Buffer Head Pointer register (register DWord 1) are written to this location either as result of an MI_REPORT_HEAD instruction or as the result of an “automatic report” (see RINGBUF registers).
Fh:5h	**Reserved.** Must not be used.
10h-1Bh	**Context Status DWords.**
1Ch-1Eh	**Reserved.** Must not be used.
1Fh	**Last Written Status Offset.**
20h-3FFh	These locations can be used for general purpose via the MI_STORE_DATA_INDEX or MI_STORE_DATA_IMM instructions.

1.1.2.2 PP_DIR_BASE – Page Directory Base Register

PP_DIR_BASE – Page Directory Base Register	
Register Type:	MMIO_CS
Address Offset:	{DevSNB] Write offset: 0x2228 Read offset: 0x2518
Project:	All
Default Value:	0000 0000h
Access:	R/W
Size (in bits):	32
This register contains the offset into the GGTT where the (current context’s) PPGTT page directory begins. This register is restored with context. The Page Directory Base Address is set by SW only by modifying the value of this register in the context image such that the new value is restored the next time the context runs. A write via MMIO to this register triggers the render pipe to fetch all PDs. **Programming Note:** The **MBC Driver Boot Enable** bit in **MBCTL** register must be set *before* this register is written to upon boot up (including S3 exit)	
Bit	**Description**

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

1.1.2.3 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:	{DevSNB] Write only. Cannot read via MMIO
Size (in bits):	64

This register controls update of the on-chip PPGTT Directory Cache during a context restore. Bits that are set will trigger the load of the corresponding 16 directory entry group. This register is restored with context (prior to restoring the on-chip directory cache itself). This register is also restored when switching to a context whose LRCA matches the current CCID if the **Force PD Restore** bit is set in the context descriptor.

The context image of this register must be updated and maintained by SW; SW should not normally need to read this register.

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

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

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.

1.1.3.1 PRBL_SF – Probe List Slot Fault Register

PRBL_SF – Probe List Slot Fault Register	
Register Type:	MMIO_CS
Address Offset:	2680h {DevSNB]
Project:	All
Default Value:	0000 0000h
Access:	RO
Size (in bits):	64

This register contains the fault bits for the probe slots, one bit for each cacheline of the 1024 probe slot memory area. It cannot be directly written by SW. The image of this register in the per-process HW status page can be read after a context switch (due to surface fault) to determine which cachelines of the probe list contain faulting probes. This register is saved with context. It is not restored but recomputed while re-validating the probe list on a context restore.

Bit	Description
63:0	**Slot Fault Line 63:0** Project: All Format: Array:Enable If set, indicates that the corresponding probe list cacheline (in memory) contains a probe that has faulted.

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.

1.1.3.2 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	Description
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).

1.1.3.3 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	Description
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

1.1.4 Context Save Registers

1.1.4.1 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	Description
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

1.1.4.2 BB_START_ADDR—Batch Buffer Start Head Pointer Register

BB_START_ADDR—Batch Buffer Start Head Pointer Register	
Register Type:	MMIO_CS
Address Offset:	2150h
Project:	All
Default Value:	0000 0000 0000 0000h
Access:	RO
Size (in bits):	32

This register contains the address specified in the last MI_START_BATCH_BUFFER command.

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

Bit	Description
31:2	**Batch Buffer Start Head Pointer** Project: All Format: GraphicsAddress[31:2] This field specifies the DWord-aligned Graphics Memory Address where the last initiated Batch Buffer starting address.
1:0	**Reserved** Project: All Format: MBZ

1.1.4.3 BB_OFFSET—Batch Buffer Address Difference Register

BB_ADDR_DIFF—Batch Address Difference Register	
Register Type:	MMIO_CS
Address Offset:	2154h
Project:	All
Default Value:	0000 0000 0000 0000h
Access:	RO
Size (in bits):	32

This register contains the difference between the start of the last batch and where the last initiated Batch Buffer is currently fetching commands.

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

Bit	Description
31:2	**Batch Buffer Address Difference** Project: All Format: GraphicsAddress[31:2] This field specifies the DWord-aligned difference between the starting address of the batch buffer and where the last initiated Batch Buffer is currently fetching commands.
1:0	**Reserved** Project: All Format: MBZ

1.1.5 Mode and Misc Ctrl Registers

1.1.5.1 MI_MODE — Mode Register for Software Interface

MI_MODE — Mode Register for Software Interface

Register Type: MMIO_CS
Address Offset: 209Ch
Project: All
Default Value: 00000000h
Access: R/W
Size (in bits): 32

The MI_MODE register contains information that controls software interface aspects of the Memory Interface function.

Bit	Description
31:16	**Masks** Format: Mask[15:0] A “1” in a bit in this field allows the modification of the corresponding bit in Bits 15:0
15	**Suspend Flush** Project: DevSNB Default Value: 0h DefaultVaueDesc Format: U1 FormatDesc

Value	Name	Description	Project
0h	No Delay	HW will not delay flush, this bit will get cleared by MI_SUSPEND_FLUSH as well	All
1h	Delay Flush	HW will delay the flush because of sync flush or VTD regimes until reset, this bit will get set by MI_SUSPEND_FLUSH as well	All

Programming Notes	Project
This should only be written to from the ring using MI_SUSPEND_FLUSH. It is considered undefined if written by software through MMIO	All

MI_MODE — Mode Register for Software Interface

14 **Async Flip Performance mode**

Project: All

Default Value: 0h

Format: U1

[DevSNB] This bit must be set to '1'

Value	Name	Description	Project
0h	Performance mode enabled	The stall of the flip event is in the windower	All
1h	Performance mode disabled	The stall of the flip event is in the command stream	All

13 **Flush Performance mode**

Project: All

Default Value: 0h

Format: U1

Value	Name	Description	Project
0h	run fast restore	No NonPipelined SV flush.	All
1h	run slow legacy restore	With NonPipelined SV flush.	All

12 **MI_FLUSH Enable**

Project: DevSNB

Default Value: 0h DefaultVaueDesc

Format: Enable

PIPE_CONTROL is a superset of MI_FLUSH. Since MI_FLUSH is redundant, it will be removed in future projects beyond GT. By default, it is disabled

Value	Name	Description	Project
0h	Disable	If an MI_FLUSH is parsed with this bit disabled, the parser will stall and the parser error bit will be set in the ESR creating an interrupt	DevSNB
1h	Enable	If an MI_FLUSH is parsed with this bit enabled, the parser will execute the legacy command according to the bspec	DevSNB

11 **Invalidate UHPTR enable** Project: All Format: Enable

If bit set H/W clears the valid bit of UHPTR (2134h, bit 0) when current active head pointer is equal to UHPTR.

MI_MODE — Mode Register for Software Interface

10	**Reserved** Project: All Format: MBZ
9	**Rings Idle** Project: All Default Value: 0h Format: U1 Read Only Status bit

Value	Name	Description	Project
0h	Not Idle	Parser not Idle or Ring Arbiter not Idle.	All
1h	Idle	Parser Idle and Ring Arbiter Idle.	All

Programming Notes	Project
Writes to this bit are not allowed.	All

8	**Stop Rings** Project: All Default Value: 0h Format: U1

Value	Name	Description	Project
0h		Normal Operation.	All
1h		Parser is turned off and Ring arbitration is turned off.	All

Programming Notes	Project
Software must set this bit to force the Rings and Command Parser to Idle. Software must read a “1” in Ring Idle bit after setting this bit to ensure that the hardware is idle.	All
Software must clear this bit for Rings to resume normal operation.	All

7	**Vertex Shader Cache Mode** Project: All Default Value: 0h Format: U1

Value	Name	Description	Project
0h	Non-LRA	Non-LRA mode of allocation. Vertex shader cache is allocated on the basis of the reference count of individual vertices	All
1h	LRA	LRA mode of allocation. Used for validation purposes.	All

MI_MODE — Mode Register for Software Interface

6 — Vertex Shader Timer Dispatch Enable

Project: All

Default Value: 0h

Format: Enable

Value	Name	Description	Project
0h	Disable	Disable the timer for dispatch of single vertices from the vertex shader. Vertex shader will try to collect 2 vertices before a dispatch	All
1h	Enable	Enable the timer for dispatch of single vertices. Dispatch a single vertex shader thread after the timer expires.	All

Programming Notes	Project
To avoid deadlock conditions in hardware this bit needs to be set for normal operation.	All

5 — Reserved Project: All Format: MBZ

4 — Enable Software Element Configuration

Project: DevSNB+

Default Value: 0h

Format: Enable

Value	Name	Description	Project
0h	Disable	Hardware will choose how to pass elements down the quad pipe of the Vertex Fetch	DevSNB
1h	Enable	Software will be able to choose which configuration to pass elements down the Vertex Fetch pipeline. See the 3D_VERTEX_ELEMENTS command in the 3D_pipeline chapter for more details.	DevSNB

3:1 — Reserved Project: All Format: MBZ

Read/Write

0 — Mask IIR disable Project: All Format: Disable

Mask IIR disable. Nominally the Interrupt controller masks interrupts in the IIR register if an interrupt acknowledge from the 3gio interface is pending. Setting this bit to a “1” allows interrupts to be visible to the interrupt controller while an interrupt acknowledge is pending.

1.1.5.2 GFX_MODE – Graphics Mode Register

GFX_MODE

Register Type: MMIO
Address Offset: 2520h {DevSNB]
Project: All
Default Value: 00000800h {DevSNB]

Access: R/W
Size (in bits): 32
Trusted Type: 1

This register contains a control bit for the 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)
15	**Reserved**
14	**Reserved** Project: All Format: MBZ
13	**Flush TLB invalidation Mode** Project: All Format: U1 This field controls the invalidation if the TLB cache inside the hardware. When *enabled* this bit limits the invalidation of the TLB only to batch buffer boundaries or to pipe_control commands which have the TLB invalidation bit set. If *disabled*, the TLB caches are flushed for every full flush of the pipeline. [DevSNB A] This bit must be '0'
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)

Value	Name	Description	Project
0h	Disabled	when '0', the TLB caches are flushed for every full flush of the pipeline.	{DevSNB]
1h	Enabled	when '1' only send TLB inv on batch buffer boundaries or when PIPE_CONTROL w/ TLB inv bit is set	{DevSNB]

GFX_MODE

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	Name	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-cmdbuffer 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 **Reserved**

GFX_MODE

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 single-context scheduling 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

Programming Notes:

- **[DevSNB A/B {W/A}]:** If RC6 is enabled and PPGTT mode is used, software must program the **CTX_WA_PTR (0x2058)** on boot for power context. Inside the work-around batch memory, there must be several MI_LOAD_REGISTER_IMM (LRI) commands to reload the PD Offset.
  - LRI address = 0x02228, data = Render PD base addr (statically defined)
  - LRI address = 0x12228, data = MFX PD base addr (statically defined)
  - LRI address = 0x22228, data = Blitter PD base addr (statically defined)
- [**DevSNB]** PPGTT memory writes by MI_* (such as MI_STORE_DATA_IMM) and PIPE_CONTROL are not supported.

8 **Reserved** Project:

7:0 **Reserved** Project: All Format: MBZ

1.1.5.3 GT_MODE – GT Mode Register [DevSNB+]

GT_MODE – GT Mode Register

Register Type: MMIO_CS[DevSNB]

Address Offset: 20D0h[DevSNB]

Project: DevSNB+
Default Value: 00000000h
Access: R/W
Size (in bits): 32
Trusted Type: 1

This Register is used to control the 6EU and 12EU configuration for SNB.
Write 0x01FF01FF to this register enables the 6EU mode.
[DevSNB A] Software must perform a read-modify-write sequence to update register after initial value is written. Also, every write must have value [31:16] = 0xFFFF

Bit	Description
31:16	**Mask Bits** Format: Mask[15:0] Must be set to modify corresponding bit in Bits 15:0. (All implemented bits)
15	**Reserved**
14:11	**Reserved** :
14:13	**Reserved**
14:13	**Reserved** Project: Format: MBZ
12:11	**Reserved**
10	**Reserved**
9	**WIZ Hashing Mode High Bit** Project: DevSNB-B+ Default Value: 1h Format: U1 This field adds additional hashing modes in combination with the **WIZ Hashing Mode** field. The Value column in the table below refers to this field (high bit) and the **WIZ Hashing Mode** field (low bit). *This field is don't care if the Hashing Disable bit is set.*

Value	Name	Description	Project
0h		8x8 Checkerboard hashing	DevSNB-B+
1h		8x4 Checkerboard hashing	DevSNB-B+
2h		16x4 Checkerboard hashing	DevSNB-B+
3h		Reserved	

GT_MODE – GT Mode Register

8 — Full Rate Sampler Disable

Project: DevSNB
Default Value: 0h
Format: Enable

This field configures the sampler rate.

Value	Name	Description	Project
0h	Disable	Full rate sampler	DevSNB
1h	Enable	Half rate sampler	DevSNB

7 — WIZ Hashing Mode

Project: DevSNB+
Default Value: 0h
Format: U1

This field configures the Hashing mode in Windower. For [DevSNB-B+], the **WIZ Hashing Mode High Bit** field is combined with this field to enable additional modes.

This field is don't care if the Hashing Disable bit is set.

Value	Name	Description	Project
0h		16x4 Checkerboard hashing	DevSNB
1h		8x4 Checkerboard hashing	DevSNB

6 — Reserved

5 — TD Four Row Dispatch Disable

Project: DevSNB
Default Value: 0h
Format: Enable

This field configures the number of rows TD dispatchs thread into.

Value	Name	Description	Project
0h	Disable	TD dispatchs to all 4 rows	DevSNB
1h	Enable	TD dispatchs to only row0 and row1	DevSNB

4 — Full Size URB Disable

Project: DevSNB
Default Value: 0h
Format: Enable

This field configures the size of the URB.

Value	Name	Description	Project
0h	Enable	Full size URB	DevSNB
1h	Disable	Half size URB	DevSNB

GT_MODE – GT Mode Register

3 — **Full Size SF FIFO Disable**

Project: DevSNB
Default Value: 0h
Format: Enable
This field configures the size of the FIFO between SF and PSD.

Value	Name	Description	Project
0h	Enable	Full size SF FIFO	DevSNB
1h	Disable	Half size SF FIFO	DevSNB

2 — **Reserved** Project: All Format: MBZ

1 — **VS Quad Thread Dispatch Disable**

Project: DevSNB
Default Value: 0h
Format: Enable
This field configures the number of dispatch ports in VS unit.

Value	Name	Description	Project
0h	Enable	Quad thread dispatch enabled for VS	DevSNB
1h	Disable	Quad thread dispatch disabled for VS	DevSNB

0 — **Reserved**

1.1.5.4 Cache_Mode_0— Cache Mode Register 0

Cache_Mode_0— Cache Mode Register 0

Register Type:	MMIO_CS[DevSNB]
Address Offset:	2120h [DevSNB]
Project:	All
Default Value:	0000 6820h [DevSNB]
Access:	R/W
Size (in bits):	32

This register is used to control the operation of the Render and Sampler L2 Caches. All reserved bits are implemented as read/write.

Before changing the value of this register, GFX pipeline must be idle i.e. full flush is required

This Register is saved and restored as part of Context.

Bit	Description
31:16	**Masks** Format: Mask[15:0] A “1” in a bit in this field allows the modification of the corresponding bit in Bits 15:0.
15	**Sampler L2 Disable** Project: All Default Value: 0h Format: Disable
14:10	**Reserved** Project: All Format: MBZ

Value	Name	Description	Project
0h		Sampler L2 Cache Enabled.	All
1h		Sampler L2 Cache Disabled all accesses are treated as misses.	All

Errata	Description	Project
BWT012	Setting this bit is UNDEFINED.	DevBW-A,B

Cache_Mode_0— Cache Mode Register 0

9 — Sampler L2 TLB Prefetch Enable

Project: All
Default Value: 0h
Format: Enable

Value	Name	Description	Project
0h		TLB Prefetch Disabled	All
1h		TLB Prefetch Enabled	All

8 — Reserved

7:6 — Sampler L2 Request Arbitration

Project: All
Default Value: 0h
Format: U2

Value	Name	Description	Project
00		Round Robin	All
01		Fetch are Highest Priority	All
10		Constants are Highest Priority	All
11		Reserved	All

5 — STC Eviction Policy Project: All Format: Disable

If this bit is set, STCunit will have LRA as replacement policy. The default value i.e. (when this bit is reset) indicates that non-LRA eviction policy. This bit must be reset. LRA replacement policy is not supported.

4 — RCC Eviction Policy Project: [DevSNB+] Format: Disable

If this bit is set, RCCunit will have LRA as replacement policy. The default value i.e. (when this bit is reset) indicates that non-LRA eviction policy. This bit must be reset. LRA replacement policy is not supported.

3 — Reserved

2 — Reserved Project: Format: MBZ

2 — Reserved

1 — Disable clock gating in the pixel backend Project: All Format: Disable

MCL related clock gating is disabled in the pixel backend.
Before setting this bit to 1, the instruction/state caches must be invalidated. [DevSNB:{WKA}]

Cache_Mode_0— Cache Mode Register 0

0	**Render Cache Operational Flush Enable** Project: [All] Default Value: 0h Format: Enable

Value	Name	Description	Project
0h	Disable	Operational Flush Disabled (recommended for performance when not rendering to the front buffer)	All
1h	Enable	Operational Flush Enabled (required when rendering to the front buffer)	All

Errata	Description	Project
	This bit must be 0. Operational Flushes are not supported in [DevSNB]. SW must flush the render target after front buffer rendering.	[DevSNB]

1.1.5.5 Cache_Mode_1— Cache Mode Register 1

Cache_Mode_1— Cache Mode Register 1

Register Type: MMIO_CS [DevSNB]

Address Offset: 2124h [DevSNB]

Project: All
Default Value: 0000 0180h
Access: Read/32 bit Write
Size (in bits): 32

Before changing the value of this register, GFX pipeline must be idle i.e. full flush is required.

This Register is saved and restored as part of Context.

Bit	Description
31:16	**Mask Bits for 15:0** Format: Mask[15:0] Must be set to modify corresponding data bit. Reads to this field returns zero.
15	**Reserved** Project: All Format: MBZ

Cache_Mode_1— Cache Mode Register 1

14	**Reserved**
13	**Reserved**
12	**HIZ Eviction Policy** Project: All Default Value: 0h Format: U1 If this bit is set, Hizunit will have LRA as replacement policy. The default value i.e. (when this bit is reset) indicates the non-LRA eviction policy. For performance reasons, this bit must be reset.

Value	Name	Description	Project
0h		Non-LRA eviction Policy	All
1		LRA eviction Policy	All

11	**DAP Instruction and State Cache Invalidate** Project: All Default Value: 0h Format: U1 When this field is set, DAP instruction and state caches (level 1 and level 2) are invalidated.

Value	Name	Description	Project
0h		Normal Cache operation.	All
1		Reserved	All

10	**Instruction Level 1 Cache and In-Flight Queue Disable** Project: All Default Value: 0h Format: Disable

Value	Name	Description	Project
0h		Cache is enabled.	All
1h		Reserved	All

Cache_Mode_1— Cache Mode Register 1

9 **Instruction and State Level 2 Cache Fill Buffers Disable**

Project: All

Default Value: 0h

Format: Disable

Value	Name	Description	Project
0h		Fill Buffers are enabled.	All
1h		Reserved	All

8:7 **Sampler Cache Set XOR selection**

Project: All

Default Value: 3h

Format: U2

These bits have an impact only when the Sampler cache is configured in 16 way set associative mode. If the cache is being used for immediate data or for blitter data these bits have no effect.

Value	Name	Description	Project
00	Default value	Default behavior to calculate set address, no XOR.	All
01	Scheme 1	New_set_mask[3:0] = Tiled_address[16:13] New_set[3:0] <= New_set_mask[3:0] ^ Old_set[3:0] Rationale: These bits can distinguish among 16 different equivalent classes of virtual pages. These bits also represent the lsb for tile rows ranging from a pitch of 1 tile to 16 tiles.	All
10	Scheme 2	New_set_mask[3] = Tiled_address[17] ^ Tiled_address[16] New_set_mask[2] = Tiled_address[16] ^ Tiled_address[15] New_set_mask[1] = Tiled_address[15] ^ Tiled_address[14 New_set_mask[0] = Tiled_address[14] ^ Tiled_address[13] New_set[3:0] <= New_set_mask[3:0] ^ Old_set[3:0] Rationale: More bits on each XOR can give better statistical uniformity on sets and since two lsbs are taken for each tile row size, it reduces the chance of aliasing on sets.	All
11	Scheme 3	New_set_mask[3] = Tiled_address[22] ^	All

			Tiled_address[21] ^ Tiled_address[20] ^ Tiled_address[19] New_set_mask[2] = Tiled_address[18] ^ Tiled_address[17] ^ Tiled_address[16] New_set_mask[1] = Tiled_address[15] ^ Tiled_address[14] New_set_mask[0] = Tiled_address[13] New_set[3:0] <= New_set_mask[3:0] ^ Old_set[3:0] Rationale: More bits on each XOR can give better statistical uniformity on sets and since each XOR has different bits, it reduces the chance of aliasing on sets even more.	

6:5	**Reserved**	Project:	DevSNB	Format:

4 **Data Cache Disable**

Project: All

Default Value: 0h

Format: Disable

Value	Name	Description	Project
0h		Cache is enable	All
1h		Reserved	All

3 **Depth Read Hit Write-Only Optimization Disable**

Project: DevSNB

Default Value: 0h

Format: Disable

Value	Name	Description	Project
0h		Read Hit Write-only optimization is enabled in the Depth cache (RCZ).	DevSNB+
1h		Read Hit Write-only optimization is disabled in the Depth cache (RCZ).	DevSNB+

Cache_Mode_1— Cache Mode Register 1

2

Depth Cache LRA Hunt Feature Disable

Project: DevSNB

Default Value: 0h

Format: Disable

Value	Name	Description	Project
0h		LRA Hunt eviction policy is enabled for Depth Cache (RCZ).	DevSNB+
1h		LRA Hunt eviction policy is disabled. In this case, strict LRA eviction policy is used in Depth Cache(RCZ).	DevSNB+

1

Instruction and State Level 2 Cache Disable

Project: All

Default Value: 0h

Format: Disable

ISC cache must be invalidated before toggling this bit.

Value	Name	Description	Project
0h		Cache is enabled.	All
1h		Reserved	All

0

Instruction Level 1 Cache Disable

Project: All

Default Value: 0h

Format: Disable

Value	Name	Description	Project
0h		Cache is enabled.	All
1h		Reserved	All

1.1.5.6 INSTPM—Instruction Parser Mode Register

INSTPM—Instruction Parser Mode Register	
Register Type:	MMIO_CS
Address Offset:	20C0h
Project:	All
Default Value:	00006000h
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	Description
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	**Reserved** Project: All Format: MBZ
14:13	**Reserved**
12	**Reserved**
11	**CLFLUSH Toggle** Project: DevSN B+ Format: U1 This bit changes polarity each time the MI_CLFLUSH command completes.
10	**Reserved** Project: All Format: MBZ
9	**TLB Invalidate** Project: DevSNB+ Format: U1 If set, this bit allows the command stream engine to invalidate the 3D render TLBs. This bit is valid only with the Sync flush enable. ***Note: GFX soft resets do not invalidate TLBs, it is upto GFX driver to explicitly invalidate TLBs post reset.***

INSTPM—Instruction Parser Mode Register	
8	**Memory Sync Enable** Project: DevSNB+ Format: U1 If set, this bit allows the command stream engine to write out the data from the local caches to memory. This bit is valid only with the Sync flush enable
7	**Force Sync Command Ordering** Project: DevSNB+ Format: Enable By default, driver/OS synchronization commands (MI_STORE_DATA_IMM, for instance) can execute out of order with respect to 3D state and 3D primitive commands. When set, this bit forces ordering of these command. See section 3.2.2 for a list of these commands
6	**CONSTANT_BUFFER Address Offset Disable** Project: All Format: U1 When this bit is clear, the 3DSTATE_CONSTANT_* Buffers' Starting Address is used as a DynamicStateOffset. I.e., it serves as an offset from the Dynamic State Base Address **[DevSNB+]**. Accesses will be subject to Dynamic State bounds checking. 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
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**. • Sync flush in multi-context scheduling mode can be used only if there is one context in hardware and no new contexts can be scheduled till sync flush is complete. • Software is expected to follow restriction above or not use Sync flush in multi-context scheduling mode Format = Enable (cleared by HW) DevSNB{WA: D2}: If 0x21d0[7] = '1', the following work-around is needed Write 0x2054[31:0] = 0x000FFFF <-- Set the idle counter to max value Write 0x2700[31:0] = 0x00000000 <-- Wake up CS (but don't do anything) Poll 0x22AC[3:0] = 0 <-- Guarantees render pipe is awake Write 0x2050[31:0] = 0x00010001 <-- disable sequence VT-d request(Sync Flush) <-- Normal VT-d cycles(Replace with Sync Flush Steps) Write 0x2054[31:0] = <old value> <-- Set to value before flow began Write 0x2050[31:0] = 0x00010000 <-- Enable sequence (to enter RC6)
4	**Reserved** Project: All

INSTPM—Instruction Parser Mode Register	
3	**Media Instruction Disable** Project: All Format: U1 This bit instructs the Renderer instruction parser to parse and error-check Media 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

1.1.5.7 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	Description
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).

1.1.5.8 FBC RT BASE ADDRESS REGISTER

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

Bit	Description
31:12	4KB aligned Base Address as mapped in the PPGTT (in the multi-context scheduling mode) OR in the GGTT (in the single-context scheduling mode) For the render target. This register must be programmed in either multi-context scheduling or single-context scheduling mode. 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: All 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.	All
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.	All

FBC_RT_BASE_ADDR_REGISTER

0	**PPGTT Render Target Base Address Valid for FBC** Project: All Security: None Access: None Exists If: Always Default Value: 0h DefaultVaueDesc Mask: MMIO(0x2000)#16 Format: Enable FormatDesc Address: GraphicsAddress[31:0] Surface Type: U32 Range 0..2^32-1 BitFieldDesc

Value	Name	Description	Project
0h		Base address in this register [31:12] is not valid and therefore FBC will not get any modifications from rendering.	
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.	

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

1.1.5.10 RBSYNC – Render/Blitter Semaphore Sync Register

RBSYNC – Render/Blitter Semaphore Sync Register	
Register Type:	MMIO_CS
Address Offset:	2044h
Project:	All
Default Value:	00000000h
Access:	R/W
Size (in bits):	32
Trusted Type:	1

This register is written by BCS, read by CS.

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

1.1.5.11 SEMA_REG—Semaphore General Sync Registers

SEMA_REG—Semaphore General Sync Registers	
Register Type:	MMIO_CS
Address Offset:	2680-26FFh
Project:	All
Default Value:	0h
Access:	R/W
Size (in bits):	32 registers x 32b

This register contains the semaphore value to be compared with the value specifed in the MI_SEMAPHORE_MBOX command. The register value in the command will be compared with the MMIO offset specifed in the table below:

Register Number	MMIO Offset
0	0x2680
1	0x2684
2	0x2688
3	0x268C

SEMA_REG—Semaphore General Sync Registers		
4	0x2690	
5	0x2694	
6	0x2698	
7	0x269C	
8	0x26A0	
9	0x26A4	
10	0x26A8	
11	0x26AC	
12	0x26B0	
13	0x26B4	
14	0x26B8	
15	0x26BC	
16	0x26C0	
17	0x26C4	
18	0x26C8	
19	0x26CC	
20	0x26D0	
21	0x26D4	
22	0x26D8	
23	0x26DC	

SEMA_REG—Semaphore General Sync Registers

24	0x26E0
25	0x26E4
26	0x26E8
27	0x26EC
28	0x26F0
29	0x26F4
30	0x26F8
31	0x26FC

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

1.1.6 RINGBUF — Ring Buffer Registers

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

1.1.6.1 RING_BUFFER_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	Description
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

1.1.6.2 RING_BUFFER_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	Description
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

1.1.6.3 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	Description
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.
11:0	**Reserved** Project: All Format: MBZ

1.1.6.4 RING_BUFFER_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	Description
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.
10	**Semaphore Wait** Project: DevSN B+ Format: U32 Indicates that this ring has executed a MI_SEMAPHORE_MBOX instruction with register compare and is currently waiting. Software can write a “1” to clear this bit, write of “0” has no effect. When the RB is waiting for the compare to meet and this bit is cleared, the wait will be terminated and the RB will be returned to arbitration.
9:3	**Reserved** Project: All Format: MBZ

RING_BUFFER_CONTROL

2:1 **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.

Value	Name	Description	Project
0h	MI_AUTOREPORT_OFF	Automatic reporting disabled	All
1h	MI_AUTOREPORT_64KB MI_AUTOREPORT_4KB	Report every 16 pages (64KB) When the **Per-Process Virtual Address Space** bit is set, the ring buffer reports every 4KB	All
2h	Reserved	Reserved	All
3h	MI_AUTOREPORT_128KB	Report every 32 pages (128KB)	All

Programming Notes	Project
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.	All

0 **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*.

1.1.6.5 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	Description
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	Name	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

1.1.7 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.

1.1.7.1 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	Description
31	**Count Select** Project: [DevSNB] Format: select 0 – Use the timestamp to increment the watchdog count (every 640ns) 1 – Use the fixed function clock (csclk) to increment the watchdog count
30:0	**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.

1.1.7.2 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	Description
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.

1.1.7.3 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	Description
31:0	**Counter Value** Project: All Format: U32 This register reflects the render watchdog counter value itself. It cannot be written to.

1.1.8 Interrupt Control Registers

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

Table 1-1. Bit Definition for Interrupt Control Registers

Bit	Description
31:10	**Reserved. MBZ** These bits may be assigned to interrupts on future products/steppings.
9	**Performance Monitoring Buffer Half-Full Interrupt:** For internal trigger (timer based) based reporting, if the report buffer crosses half full limit, this interrupt is generated.
8	**Context Switch Interrupt:** Set when a context switch has just occurred. **Per-Process Virtual Address Space bit** needs to be set for this interrupt to occur**.**
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	**L3 Parity Error:** When this bit is set, L3 cache controller is indicating that it has encountered an parity error while checking the data.
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**
1	**Reserved**
0	**Render Command Parser User Interrupt:** This status bit is set when an MI_USER_INTERRUPT instruction is executed on the Render Command Parser. Note that instruction execution is not halted and proceeds normally. A mechanism such as an MI_STORE_DATA instruction is required to associate a particular meaning to a user interrupt.

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)
9	**Performance Monitoring Buffer Half-Full Interrupt**	Set when event occurs, cleared when event cleared
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	**Reserved**	
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
1	Reserved	
0	User Interrupt	0

1.1.8.1 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).
- At most 1 bit can be unmasked at any given time.

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

1.1.8.2 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	Description
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	Name	Description	Project
0h	Not Masked	Will be reported in the IIR	All
1h	Masked	Will not be reported in the IIR	All

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

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

The following table describes the Hardware-Detected Error bits:

Table 1-2. Hardware-Detected Error Bits

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

1.1.8.3.1 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)), except for ther unrecoverable bits described.)

Bit	Description
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. 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.

Bit	Description
31:5	Reserved: MBZ

Value	Name	Description	Project
1h	Error occurred	Error occurred	All

Programming Notes	Project
Writing a '1' to a set bit will cause that error condition to be cleared. However, the Page Table Error bit (Bit 4) nor the Instruction Error bit (Bit 0) cannot be cleared except by reset (i.e., it is a fatal error).	All

1.1.8.3.2 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	Description
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	Name	Description	Project
0h	Not Masked	Will be reported in the EIR	All
1h	Masked	Will not be reported in the EIR	All

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

ESR—Error Status Register

15:0

Error Status Bits

Project: All

Default Value: 0h

Format: Array of error condition bits See Table 1 5. Hardware-Detected Error Bits

This register contains the non-persistent values of all hardware-detected error condition bits.

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

1.1.9 Logical Context Support

1.1.9.1 BB_ADDR—Batch Buffer Head Pointer Register

BB_ADDR—Batch Buffer Head Pointer Register

Register Type: MMIO_CS
Address Offset: 2140h
Project: All
Default Value: 0000 0000 0000 0000h
Access: RO
Size (in bits): 32

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

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

Bit	Description
31:2	**Batch Buffer Head Pointer** Project: All Format: GraphicsAddress[31:2] This field specifies the DWord-aligned Graphics Memory Address where the last initiated Batch Buffer is currently fetching commands. If no batch buffer is currently active, the Valid bit will be 0 and this field will be meaningless.
1	**Reserved** Project: All Format: MBZ
0	**Valid** Project: All Default Value: 0h Format: U1 **Value** / **Name** / **Description** / **Project** 0h / Invalid / Batch buffer Invalid / All 1h / Valid / Batch buffer Valid / All

1.1.9.2 BB_STATE – Batch Buffer State Register

BB_STATE – Batch Buffer State Register

Register Type: MMIO_CS
Address Offset: 2110h
Project: All
Default Value: 0000 0000h
Access: RO
Size (in bits): 32

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

This register should *not* be written by software directly. Software should always set these fields via the MI_BATCH_BUFFER_START command when initiating a batch buffer.

{DevSNB] This register is *not* restored with context. As a consequence, MI_WAIT_FOR_EVENT cannot enter RC6 inside a batch buffer with any of these attributes set

Bit	Description
31:8	**Reserved** Project: All Format: MBZ
7	**Reserved** Project: Format:
6	**Clear Command Buffer Enable** Project: All Format: U1 If set the batch buffer is getting executed from the Write Once protected memory area. The address of the batch buffer is an offset into the WOPCM area.
5	**Buffer Security Indicator** Project: All Default Value: 0h Format: MI_BufferSecurityType If set, this batch buffer is non-secure and cannot execute privileged commands nor access privileged (GGTT) memory. It will be accessed via the PPGTT. If clear, this batch buffer is secure and will be accessed via the GGTT. Note: This field reflects the effective security level and may not be the same as the Buffer Security Indicator written using MI_BATCH_BUFFER_START.
4	**Reserved**
3:0	**Reserved** Project: All Format: MBZ

Value	Name	Description	Project
0h	MIBUFFER_SECURE	Located in GGTT memory	All
1h	MIBUFFER_NONSECURE	Located in PPGTT memory	All

1.1.9.3 CTXT_SR_CTL – Context Save/Restore Control Register

CTXT_SR_CTL – Context Save/Restore Control Register					
Register Type:	2714h [DevSNB]				
Address Offset:	2714h				
Project:	All				
Default Value:	0000 0000h				
Access:	R/W				
Size (in bits):	32				
This register is saved and restored with context.					
Bit	**Description**				
31:2	**Reserved** Project: All Format: MBZ				
1	**Reserved**				
0	**Render Context Restore Inhibit** Project: All Format: U1 This is not a true register bit. This bit should be set in the context image of a ring context that is being submitted for the first time. Setting this bit will inhibit the restoring of render context (including extended context if applicable) so that restoring of an uninitialized render context can be prevented. This bit will always be set on a context save (since the render context cannot be uninitialized on context save – it will always contain at least default values.)				

1.1.9.4 CCID—Current Context Register

CCID—Current Context Register	
Register Type:	MMIO_CS
Address Offset:	2180h
Project:	All
Default Value:	0000 0000h
Access:	R/W
Size (in bits):	32
This register contains the current “logical rendering context address” associated with the ring buffer. **Programming Note:** The CCID register must not be written directly (via MMIO) unless the Command Streamer is completely idle (i.e., the Ring Buffer is empty and the pipeline is idle). Note that, under normal conditions, the CCID register should only be updated from the command stream using the MI_SET_CONTEXT command.	

CCID—Current Context Register

Bit	Description
31:12	**Logical Render Context Address (LRCA)** Project: All Default Value: 0h Address: GraphicsAddress[31:11] This field contains the 4 KB-aligned Graphics Memory Address of the current Logical Rendering Context. Bit 11 MBZ. This register will point to a Logical Pipeline Context (a subset of a Logical Rendering Context) if loaded using MI_SET_CONTEXT.
11:10	**Reserved** Project: All Format: MBZ
9	**Reserved**
8	**Reserved** Project: All Format: Must be '1'
7:4	**Reserved** Project: All Format: MBZ
3	**Extended State Save Enable** Project: All Format: Enable If set, the extended state identified in the Logical Context Data section of the Memory Data Formats chapter, is saved as part of switching away from this logical context.
2	**Extended State Restore Enable** Project: All Format: Enable If set, the extended state identified in the Logical Context Data section of the Memory Data Formats chapter, was loaded (or restored) as part of switching to this logical context.
0	**Valid** Project: All Default Value: 0h Format: U1

Value	Name	Description	Project
0h	Invalid	The other fields of this register are invalid. A switch away from the context will not invoke a context save operation.	All
1h	Valid	The other fields of this register are valid, and a switch from the context will invoke the normal context save/restore operations.	All

1.1.9.5 CXT_SIZE—Context Sizes

CXT_SIZE—Context Sizes

Register Type:	MMIO_CS
Address Offset:	Write: 21A8h, Read: 21A0h
Project:	DevSNB
Default Value:	1E0CDDD3h
Access:	Read/32 bit Write
Size (in bits):	32

Bit	Description
31:30	**Reserved** Project: All Format: MBZ
29:24	**Power Context Size** Project: Dev SNB Default Value: 1Eh DefaultVaueDesc Format: U32 FormatDesc BitFieldDesc
23:18	**Ring Context Size** Project: Dev SNB Default Value: 3h DefaultVaueDesc Format: U32 FormatDesc BitFieldDesc
17:12	**Render Context Size** Project: Dev SNB Default Value: Dh DefaultVaueDesc Format: U32 FormatDesc BitFieldDesc
11:6	**Extended Context Size** Project: Dev SNB Default Value: 37h DefaultVaueDesc Format: U32 FormatDesc BitFieldDesc
5:0	**3D Pipeline State Context Size** Project: Dev SNB Default Value: 13h DefaultVaueDesc Format: U32 FormatDesc BitFieldDesc

1.1.9.6 CXT_PIPESTATEBASE — Pipeline State Base Address

CXT_PIPESTATEBASE — Pipeline State Base Address

Register Type: MMIO_CS
Address Offset: 21B0h
Project: DevSNB
Default Value: 00000000h
Access: R/W
Size (in bits): 32

This register contains the base address where the pipeline state data is saved when PSMI interruption granularity in GFX_MODE is set to mid-triangle

Bit	Description
31:12	**Pipeline State Base Address** Project: All Default Value: 0h Invalid base address Format: Address Page Base Address The page aligned base address for pipelined state context data. **Programming Notes** • There must be 4 contiguous pages allocated with this base address to support 8 pipeline state specific context data
11:1	**Reserved** Project: All Format: MBZ
0	**Valid** Project: All Format: Bool Valid bit for 31:12. Defaults to invalid (clear)

1.1.9.7 MTCH_CID_RST – Matched Context ID Reset Register

MTCH_CID_RST – Matched Context ID Reset Register

Register Type:	MMIO_CS
Address Offset:	2524h
Project:	All
Default Value:	0000 0002h
Access:	R/W
Size (in bits):	32

This register is used to generate a Context ID specific reset (Render Only). To initiate a reset, the register is written with the pending bit set. Hardware compares the current context ID with the register and on match generates a Render Only reset. After reset is complete, HW clears the pending bit and can be programmed to generate an interrupt. The match bit is set. If the current context ID does not match this register, the pending bit is reset and an interrupt is generated. The match bit is reset.

The match indicates the result of the last comparison, and its valid only when pending bit is zero.

Please see MCIDRST interrupt bit assignment in the Interrupt Control Registers.

Bit	Description
31:12	**Match Context ID** Project: All Format: U20 Contains the context ID to be compared with the currently running context ID.
11:2	**Reserved** Project: All Format: MBZ
1	**Match** Project: All Format: U20 This bit indicates the result of the match operation; 1 means the Current Context ID matches the Match Context ID field.
0	**Pending** Project: All Format: U20 This bit indicates that a matched context ID reset is pending. The bit should be set when the register is written (in order to have a pending MTCH_CID_RST request), and will be reset by hardware to indicate that the operation is completed (Either with a match or mismatch)

1.1.9.8 SYNC_FLIP_STATUS – Wait for event and Display flip flags Register

SYNC_FLIP_STATUS – Wait for event and Display flip flags Register

Register Type: MMIO_CS
Address Offset: 25A0h

Project: All
Default Value: 0000 0000h
Access: R/W
Size (in bits): 32

This register is the saved value of what wait for events are still valid. This register is part of context save and restore for RC6 feature..

Bit	Description
31	**Reserved** Project: All Format: MBZ
30	**Display Plane A Asyncronous Display Flip Pending** 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*.
29	**Display Plane A Syncronous Flip Display Pending** 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*.
28	**Display Sprite A Syncronous Flip Display Pending** Project : All 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*.
27	**Reserved** Project: All Format: MBZ

SYNC_FLIP_STATUS – Wait for event and Display flip flags Register					
26	**Display Plane B Asyncronous Display Flip Pending**	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*.				
25	**Display Plane B Syncronous Flip Display Pending**	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*.				
24	**Display Sprite B Syncronous Flip Display Pending**	Project :	All	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*.				
23	**Display Plane A Asyncronous Performance 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*.				
22	**Display Plane A Asyncronous 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*.				
21	**Display Plane A Syncronous 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*.				

SYNC_FLIP_STATUS – Wait for event and Display flip flags Register	
20	**Display Sprite A Syncronous Flip Pending Wait Enable** Project : All 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*.
19	**Reserved** Project: All Format: MBZ
18	**Display Pipe A Scan Line Wait Enable** Project : All Format: Enable This field enables a wait while a Display Pipe A “Scan Line” condition exists. This condition is defined as the the start of the scan line specified in the Pipe A Display Scan Line Count Range Compare Register. See Scan Line Event in the Device Programming Interface chapter of *MI Functions*.
17	**Display Pipe A Vertical Blank Wait Enable** Project : All Format : U32 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 Pipe A vertical blank period. Note that this can cause a wait for up to an entire refresh period. See Vertical Blank Event (See *Programming Interface*).
16	**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*.
15	**Display Plane B Asyncronous Performance 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*.
14	**Display Plane B Asyncronous 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*.

SYNC_FLIP_STATUS – Wait for event and Display flip flags Register	
13	**Display Plane B Syncronous 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*.
12	**Display Sprite B Syncronous Flip Pending Wait Enable** Project : All 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*.
11	**Reserved** Project: All Format: MBZ
10	**Display Pipe B Scan Line Wait Enable** Project : All Format: Enable This field enables a wait while a Display Pipe B “Scan Line” condition exists. This condition is defined as the the start of the scan line specified in the Pipe B Display Scan Line Count Range Compare Register. See Scan Line Event in the Device Programming Interface chapter of *MI Functions*.
9	**Display Pipe B Vertical Blank Wait Enable** Project : All Format : U32 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*).
8	**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*.
7:5	**Reserved** Project: All Format: MBZ

SYNC_FLIP_STATUS – Wait for event and Display flip flags Register

4:0

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
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.

1.1.10 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.

1.1.10.1 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	Description
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.)

1.1.10.2 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	Description
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.)

1.1.10.3 VS_INVOCATION_COUNT— Reported Vertex Shader Invocation Counter

VS_INVOCATION_COUNT	
Register Type:	MMIO_CS
Address Offset:	2320h
Project:	All
Default Value:	00000000h; 00000000h;
Access:	R/W
Size (in bits):	64
Trusted Type:	1
This register stores the value of the vertex count shaded by VS. This register is part of the context save and restore	

Bit	Description
63:0	**VS Invocation Count Report** Number of vertex shader threads invoked by the VS stage. Updated only when **Statistics Enable** is set in VS_STATE (see the Vertex Shader Chapter in the *3D* Volume.)

1.1.10.4 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	Description
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.)

1.1.10.5 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	Description
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.)

1.1.10.6 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	Description
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.)

1.1.10.7 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	Description
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.)

1.1.10.8 PS_INVOCATION_COUNT— Reported Pixels Shaded Counter

PS_INVOCATION_COUNT	
Register Type:	MMIO_CS
Address Offset:	2348h
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 fragments that get shaded. This register is part of the context save and restore.	

Bit	Description
63:0	**PS Invocation Count** *Reflects a count of the total number of fragments that are dispatched to pixel shader invocations while **Statistics Enable** is set in the Windower. See the Windower chapter of the 3D volume for details. This count will generally be much greater than the actual count of PS threads since a single thread may process up to 32 pixels.*

1.1.10.9 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	Description
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.

1.1.10.10 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	Description
63:36	**Reserved** Project: All Format: MBZ
35:0	**Timestamp Value** Project: All Format: U32 This register toggles every 80 ns of time.

1.1.10.11 SO_NUM_PRIMS_WRITTEN— Reported Stream Output Num Primitives Written Counter

SO_NUM_PRIMS_WRITTEN— Reported Stream Output Num Primitives Written Counter	
Register Type:	MMIO_CS
Address Offset:	2288h
Project:	All
Default Value:	0000 0000 0000 0000h
Access:	R/W
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 register gets reset when write happens to register 2380h

Bit	Description
63:0	**Num Prims Written Count** Project: All 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.)

1.1.10.12 SO_PRIM_STORAGE_NEEDED — Reported Stream Output Primitive Storage Needed Counter

SO_PRIM_STORAGE_NEEDED — Reported Stream Output Primitive Storage Needed Counter	
Register Type:	MMIO_CS
Address Offset:	2280h
Project:	All
Default Value:	0000 0000 0000 0000h
Access:	RO. 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	Description
63:0	**Prim Storage Needed Count** Project: All 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 Performance Statistics Registers

[DevSNB] When an over flow condition occurs and the buffers need to be reset, or when software wants to change the OABUFFER to point to a new area in memory, Programming of the performance ring must follow the sequence below.

- Clear OA enable bit by writing 0x2360[0] = 0
- Write OASTATUS2
- Write OABUFFER
- Write OASTATUS1
- Set OA enable bit by writing 0x2360[0] = 1

1.1.11.1 OACONTROL – Observation Architecture Control

OACONTROL – Observation Architecture Control

Register Type: MMIO
Address Offset: [DevSNB] 2360h
Project: All
Default Value: 00000000h
Access: R/W
Size (in bits): 32

This register is used to program the OA unit.

Bit	Description
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.
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.
4:2	**Reserved**

Value	Name	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

OACONTROL – Observation Architecture Control

1 — Specific Context Enable

Project: [DevSNB]

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. OA unit level clock gating must be ENABLED when using specific ContextID feature.

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.

[DevSNB] When this bit is set, in order to have cohenret counts, RC6 power state must be disabled. This can be achieved by programming MMIO registers as 0xA094=0x0 and 0xA090[31]=1.

[**DevSNB]** EU clock gating must be disabled when this bit is set.

1.1.11.2 OASTATUS1 – Observation Architecture Status Register

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

This register is used to program the OA unit.

Bit	Description
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. [DevSNB+]: SW must ensure that Tail pointer and the Head Pointer (in OASTATUS2) do not have different values while programming.
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 128KB.

Value	Description	Project
0b	16KB	GEN6
1b	32KB	GEN6
2	48KB	GEN6
3	64KB	GEN6
4	80KB	GEN6
5	96KB	GEN6
6	112KB	GEN6
7	128KB	GEN6

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. [DevSNB] Erratum: This bit must be cleared after the ring is enabled and before OA is enabled.
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.

1.1.11.3 OASTATUS2 – Observation Architecture Status Register

OASTATUS2—Observation Architecture Status Register	
Register Type:	MMIO
Address Offset:	[DevSNB] 2368h
Project:	All
Default Value:	00000000h
Access:	RW
Size (in bits):	32
This register is used to program the OA unit.	
Bit	**Description**
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**.**
4:1	**Reserved** Project: All Format: MBZ

OASTATUS2—Observation Architecture Status Register	
0	**Memory select PPGTT/GGTT access** Project: All Format: U32 0 – PPGTT 1 – GGTT

1.1.11.4 OABUFFER – Observation Architecture Buffer

OABUFFER—Observation Architecture Status Register	
Register Type:	MMIO
Address Offset:	[DevSNB] 23B0h
Project:	All
Default Value:	00000000h
Access:	{DevSNB] Write Only
Size (in bits):	32
This register is used to program the OA unit. [DevSNB] This MMIO must be set *before* the **OASTATUS1** register and set *after* the OASTATUS2 register. This is to enable proper functionality of the overflow bit. [DevSNB] Report Buffer Offset Must be 512KB aligned.	
Bit	**Description**
31:6	**Report Buffer Offset** Project: All This field specifies 64B aligned GFX MEM address where the chap counter values are reported.
5	**Reserved** Project: All Format: MBZ
4	**Reserved**
3	**Reserved**

OABUFFER—Observation Architecture Status Register	
2	**OA Report Trigger Select** Project: All Format: 1 - Level Report trigger 1-2 - Edge Report trigger.
1	**Reserved**
0	**Reserved**

1.1.11.5 OASTARTTRIG1 – Observation Architecture Start Trigger

OASTARTTRIG1—Observation Architecture Buffer	
Register Type: MMIO **Address Offset:** [DevSNB] 238Ch **Project:** All **Default Value:** 00000000h **Access:** RW **Size (in bits):** 32	
This register is used to program the OA unit. **OASTARTTRIG5-8 will be used to start Boolean counters 4 to 7.** **OASTARTTRIG1-4 will be used to start Boolean counters 0 to 3.** GEN6 report trigger behavior can be derived by programming these two sets of OA START registers with the same value.	
Bit	**Description**
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

1.1.11.6 OASTARTTRIG2 – Observation Architecture Start Trigger

OASTARTTRIG2—Observation Architecture Start Trigger	
Register Type:	MMIO
Address Offset:	[DevSNB] 2388h
Project:	All
Default Value:	00000000h
Access:	RW
Size (in bits):	32

This register is used to program the OA unit.

OASTARTTRIG5-8 will be used to start Boolean counters 4 to 7.

OASTARTTRIG1-4 will be used to start Boolean counters 0 to 3.

GEN6 report trigger behavior can be derived by programming these two sets of OA START registers with the same value.

Bit	Description
31:24	**Reserved**
23	**Threshold Enable** Enable the threshold compare logic within the trigger logic.
22	**vert 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

OASTARTTRIG2—Observation Architecture Start Trigger	
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.
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.

OASTARTTRIG2—Observation Architecture Start Trigger	
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.

1.1.11.7 OAREPORTTRIG1 – Observation Architecture Report Trigger

OAREPORTTRIG1— Observation Architecture Report Trigger

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

This register is used to program the OA unit.

Bit	Description
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	Name	Description	Project
0h	Duration		All
1h	Occurence		All

1.1.11.8 OAREPORTTRIG2 – Observation Architecture Report Trigger

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

Bit	Description
31	**Report Trigger enable** **Enable the report trigger for threshold triggers**.
30:24	**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
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

1.1.11.9 CEC0-0 – Customizable Event Creation

CEC0-0— Customizable Event Creation

Register Type: MMIO

Address Offset: [DevSNB] 2390h

Project: All

Default Value: 00000000h

Access: Write Only

Size (in bits): 32

This register is used to program the OA unit.

Bit	Description
31:21	**Reserved** Project: All Format: MBZ
20:19	**Source Select** Project: All Format: U2 Selects Event for the Boolean logic. Selects the 16 bunch of events from the, Boolean Events and Previous Events.
18:3	**Compare Value** Project: All Format: U16 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.

Value	Name	Description	Project
00b	Reserved		All
01b	Prev Events	Selects the Previous events	All
10b	Boolean Events	Selects the Boolean Events	All
11b	Reserved		All

CEC0-0— Customizable Event Creation

2:0 **Compare Function** Project: All Format: U16

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.

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

1.1.11.10 CEC0-1 – Customizable Event Creation

CEC0-1— Customizable Event Creation

Register Type: MMIO
Address Offset: [DevSNB] 2394h
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	**Reserved**
15:0	**Mask** Project: All Format: U32 These 8 bits are used to mask off entries from the comparison. For each bit: 0: This bit is considered in event calculations. 1: This bit is ignored in event calculations.

1.1.11.11 CEC1-0 – Customizable Event Creation

CEC1-0— Customizable Event Creation

Register Type: MMIO
Address Offset: [DevSNB] 2398h

Project: All
Default Value: 00000000h
Access: Write Only
Size (in bits): 32

This register is used to program the OA unit.

Bit	Description
31:21	**Reserved** Project: All Format: MBZ
20:19	**Source select** Project: All Format: U2 Selects Event for the Boolean logic. Selects the 16 bunch of events from the Booleanents and Previous Events.
18:3	**Reserved** Project: All Format:
2:0	**Compare Function** Project: All Format: U3

Value	Name	Description	Project
00b	Reserved		All
01b	Prev Events	Selects the Previous events	All
10b	Boolean Events	Selects the Boolean Events	All
11b	Reserved		All

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

1.1.11.12 CEC1-1 – Customizable Event Creation

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

Bit	Description
31:16	**Considerations** Project: All Format: U32 0: The bit is considered in event calculations. 1: The bit is delayed by 1 clock before considering it in event calculations. This is particularly useful for doing state machine arc coverage
15:0	**Mask** Project: All Format: U32 These 8 bits are used to mask off entries from the comparison. For each bit: 0: This bit is considered in event calculations. 1: This bit is ignored in event calculations.

1.1.11.13 CEC2-0 – Customizable Event Creation

CEC2-0— Customizable Event Creation

Register Type: MMIO
Address Offset: [DevSNB] 23A0h

Project: All
Default Value: 00000000h
Access: Write Only
Size (in bits): 32

This register is used to program the OA unit.

Bit	Description
31:21	**Reserved** Project: All Format: MBZ
20:19	**Source select** Project: All Format: U2 Selects Event for the Boolean logic. Selects the 16 bunch of events from the Boolean Events and Previous Events. See section 5 for more details
18:3	**Compare Value** Project: All Format: U16 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.
2:0	**Compare Function** Project: All Format: U3

Value	Name	Description	Project
00b	Reserved		All
01b	Prev Events	Selects the Previous events	All
10b	Boolean Events	Selects the Boolean Events	All
11b	Reserved		All

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

1.1.11.14 CEC2-1 – Customizable Event Creation

CEC2-1— Customizable Event Creation	
Register Type:	MMIO
Address Offset:	[DevSNB] 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 0: The bit is considered in event calculations. 1: The bit is delayed by 1 clock before considering it in event calculations. This is particularly useful for doing state machine arc coverage.
15:0	**Mask** Project: All Format: U32 These 8 bits are used to mask off entries from the comparison. For each bit: 0: This bit is considered in event calculations. 1: This bit is ignored in event calculations.

1.1.11.15 CEC3-0 – Customizable Event Creation

CEC3-0—Customizable Event Creation

Register Type: MMIO
Address Offset: [DevSNB] 23A8h

Project: All
Default Value: 00000000h
Access: RW
Size (in bits): 32

This register is used to program the OA unit.

Bit	Description
31:21	**Reserved** Project: All Format: MBZ
20:19	**Source select** Project: [DevSNB] Format: U2 Selects Event for the Boolean logic. Selects the 16 bunch of events from the Boolean Events and Previous Events.
18:3	**Compare Value** Project: All Format: U16 This field is loaded to compare against the 8 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 event is asserted. This in turn can be counted by any of the CHAP counters.
2:0	**Compare Function** Project: All Format: U3

Value	Name	Description	Project
00b	Reserved		All
01b	Prev Events	Selects the Previous events	All
10b	Boolean Events	Selects the Boolean Events	All
11b	Reserved		All

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

1.1.11.16 CEC3-1 – Customizable Event Creation

CEC3-1—Customizable Event Creation	
Register Type:	MMIO
Address Offset:	{DevSNB] 23ACh
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 0: The bit is considered in event calculations. 1: The bit is delayed by 1 clock before considering it in event calculations. This is particularly useful for doing state machine arc coverage.
15:0	**Mask** Project: All Format: U32 These 8 bits are used to mask off entries from the comparison. For each bit: 0: This bit is considered in event calculations. 1: This bit is ignored in event calculations.

1.2 Memory Interface Commands for Rendering Engine

1.2.1 Introduction

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

This chapter describes MI Commands for the 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. Refer to the *Preface* chapter for product specific summary.

1.2.2 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. If set, however, mid-triangle modes like PSMI cannot be enabled.

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

1.2.3 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 Notes:

- 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 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

1.2.4 MI_ARB_ON_OFF

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

The MI_ARB_ON_OFF instruction is used to disable/enable context switching. Note that context switching will remain disabled until re-enabled through use of this command. This command will also prevent a switch in the case of waiting on events, running out of commands or a surface probe fault. These will effectively hang the device if allowed to occur while arbitration is off (context switching is disabled.)

This command should always be used as an off-on pair with the sequence of instructions to be protected from context switch between MI_ARB_OFF and MI_ARB_ON. Software must use this arbitration control with caution since it has the potential to increase the response time of the Render Engine to pre-emption requests.

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

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

1.2.5 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

1.2.6 MI_CONDITIONAL_BATCH_BUFFER_END

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

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

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

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

1.2.7 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*.

The batch buffer can be specified as secure or non-secure, determining the operations considered valid when initiated from within the buffer and any attached (chained) batch buffers. See Batch Buffer Protection in the Device Programming Interface chapter of *MI Functions*.

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.

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	**Reserved** Format: MBZ
	22:17	**Reserved** Project: All Format: MBZ
	16	Reserved Project: Format: .
	15	**Reserved** Project: Format:

	14:13	**Reserved** Project: All Format: MBZ
	12	**Reserved** Project: All Format:
	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	**Reserved** Format: MBZ
	9	**Reserved** Project: All Format: MBZ
	8	**Buffer Security and Address Space Indicator** Project: All Format: MI_BufferSecurityType When this command is executed from within a batch buffer (i.e., is a “chained” batch buffer command), this field is IGNORED and the next buffer in the chain inherits the initial buffer’s security characteristics. [DevSNB] When **Per-Process GTT Enable** is set, it is assumed that all code is in a secure environment, independent of address space. Under this condition, this bit only specifies the address space (GGTT or PPGTT). All commands are executed “as-is”

Value	Name	Description	Project
0h	GGTT	This batch buffer is secure and will be accessed via the GGTT.	All
1h	PPGTT	This batch buffer will always be accessed via the PPGTT	All

	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total - Bias
1	31:2	**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.
	1:0	**Reserved** Project: All Format: MBZ

1.2.7.1 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. For a batch buffer marked as non-secure if **Per-Process Virtual Address Space is set**, the command buffer fetches are generated using the PPGTT space.

“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 1-3. 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 products.

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

1.2.8 MI_CLFLUSH

MI_CLFLUSH

Project:	[DevSNB]	**Length Bias:**	2
Engine:	**Render**		

Flushes out the page given in the command out to system memory. This command is specific to the render engine. This command is *not* privileged.

The MI_CLFLUSH will generare zero-length cycles which look like zero-length writes which are dropped in SuperQ to optimize performance. To get the MI_CLFLUSH to the ring/LLC, the zero-length optimization should be disabled. S/W should disable the optimization via SGCM bit18 (set to "0") before pushing CLFLUSHs to CS ring and re-enable it after observing the storeDW post CLFLUSHs are complete.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 27h Store DW MI_CLFLUSH Format: OpCode
	22	**Use Global GTT** 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_CLFLUSH		
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2 Project: All
1	31:6	**Page Base Address** Project: All Address: GraphicsAddress[31:0] 4KB aligned Page Address which software requires hardware to flush to DRAM.
	11:0	**Reserved** Project: All Format: MBZ
2..n	31:0	**DW Representing ½ Cache Line** Project: All MBZ. The information given to hardware is the DW itself, not the contents. Hardware uses the DW count of the command to determine the offset from the base to flush out. The offset is ½ cache line (8 DW = 1HW) granular so for a full page, the command will need 4096 bytes / 4 bytes per DW / 8 DW per HW = 128 DW. Note that this is not possible given the 5:0 DW length. Software must split up the DWs with multiple MI_CLFLUSH commands. Example seen below 1st MI_CLFLUSH: address 11:0 = 0, header 5:0 = 0x3FE (62 - 1/2 CL) 2st MI_CLFLUSH: address 11:0 = 62*32, header 5:0 = 0x3FE (62 - 1/2 CL) 3st MI_CLFLUSH: address 11:0 = 62*64, header 5:0 = 0x3FE (62 - 1/2 CL) 4st MI_CLFLUSH: address 11:0 = 62*96, header 5:0 = 0x3FE (62 - 1/2 CL) ... etc.. until all requested cachelines are flushed.

1.2.9 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.

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.

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.

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.

MI_DISPLAY_FLIP

- Linear memory does not support asynchronous flips

5. DWord 3 (Left Eye Display Buffer Base Address) must not be set with synchronous flips or asynchronous flips. It is only allowed to be sent with stereo 3D flips

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.
	21:20	**Display (Plane) Select** Project: [DevSNB] Format: U2 FormatDesc This field selects which display plane is to perform the flip operation.
	19:8	**Reserved** Project: Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2 *For Synchronous Flips and Asynchronous Flips,*, this field must be programmed to 1h for a total length of 3.
1	31	**Reserved**
	30:16	**Reserved** Project: All Format: MBZ

Value	Name	Description	Project
0h	Display Plane A		All
1h	Display Plane B		All
2h	Display Sprite A		All
3h	Display Sprite B		All

MI_DISPLAY_FLIP

	15:6	**Display Buffer Pitch** Project: All Default Value: 0h DefaultVaueDesc Format: U10 *For Synchronous Flips and Stereo 3D 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 stereo 3D flip or direct thru mmio.
	5:1	**Reserved** Project: All Format: MBZ
	0	**Tile Parameter** Project: [DevSNB+] Default Value: 0h DefaultVaueDesc Format: Enable For Asynchronous Flips, this parameter cannot be changed. All the flips in a flip chain should maintain the same tile parameter as programmed with the last synchronous flip or direct thru mmio.

Value	Name	Description	Project
0h	Linear	For Syncronous Flips Only	All
1h	Tiled X		All

Programming Notes
Performing a synchronous or asynchronous flip will drop any previous synchronous flip that has not yet completed.

2	31:12	**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. In stereo 3D mode this is the right eye base address.
	11:3	**Reserved** Project: All Format: MBZ
	2	**Reserved**

Programming Notes
• The Display buffer must reside completely in Main Memory • This address is always translated via the *global* (rather than per-process) GTT

MI_DISPLAY_FLIP

	1:0	**Flip Type** Project: All Default Value: 00b Synchronous flip This field specifies whether the flip operation should be performed asynchronously to vertical retrace.

Value	Name	Description	Project
00b	Sync Flip	The flip will occur during the vertical blanking interval – thus avoiding any tearing artifacts.	All
01b	Async Flip	The flip will occur “as soon as possible” – and may exhibit tearing artifacts	All

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).
- **Asynch flips are Supported on X-Tiled Frame buffers only.**
- For Asynch Flips the Buffers used must be 32KB aligned.
- Asynch flips Supported on Display Planes A and B and C only

3	31:12	**Reserved**
	11:0	**Reserved** Project: All Format: MBZ

1.2.10 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

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.

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.

MI_FLUSH

	3	**Reserved** Project: All Format:
	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

	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

1.2.11 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.
- When base address of 0x180000 is added to the Register Offset, when executed will result in updating of the register in the other GT in GTB mode of operation then the GT from which this instruction is executed. When this instruction is executed by Command Streamer with COREID-0 will result in updating the register in GT with COREID-1 and vice versa, when base address of 0x180000 is added to the register offset.

The following addresses should NOT be used for LRIs

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

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

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

MI_LOAD_REGISTER_IMM

	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 If [11:8] is '1111', then this command will behave as a NOOP. Otherwise, the value is forwarded to the destination register.
	7:0	**DWord Length** Default Value: 1h Excludes DWord (0,1) Format: =n Total Length - 2
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). When base address of 0x180000 is added to the Register Offset, when executed will result in updating of the register in the other GT in GTB mode of operation then the GT from which this instruction is executed. When this instruction is executed by Command Streamer with COREID-0 will result in updating the register in GT with COREID-1 and vice versa, when base address of 0x180000 is added to the register 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.

1.2.12 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 previous products, the process time to execute a NOP command is min of 6 clock cycles.

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

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

1.2.13 Surface Probing

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

1.2.14 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.

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

1.2.15 MI_SEMAPHORE_MBOX

MI_SEMAPHORE_MBOX			
Project:	[DevSNB]	**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.

MI_SEMAPHORE_MBOX

	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
	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.
	17:16	**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: [Reserved] 2: BCS regiser (RBSYNC) 3. Use General Register Select
	15:14	**Reserved** Project: All Format: MBZ
	13:8	**Reserved** Project: **All** Format:
	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

1.2.16 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 is legal only if **Per-Process Virtual Address Space** in the GFX_MODE register is re*set*.
- 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.
- [DevSNB] If **Flush TLB invalidation Mode** is *enabled* it's the driver's responsibility to invalidate the TLBs at least once after the previous context switch after any GTT mappings changed (including new GTT entries). This can be done by a pipelined PIPE_CONTROL with TLB inv bit set immediately before MI_SET_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

MI_SET_CONTEXT

1	31:12	**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. **[DevSNB] This field needs to be 4KB aligned virtual address.**
	11:10	**Reserved** Project: All Format: MBZ
	9	**Reserved** Format: MBZ
	8	**Reserved, Must be 1** Project: All Format: Must Be One
	7:5	**Reserved** Project: All Format: MBZ
	4	Reserved
	3	**Extended State Save Enable** Project: {DevSNB] Format: U32 If set, the extended state identified in the Logical Context Data section of the Memory Data Formats chapter is saved as part of switching away from this logical context. This bit will be stored in the associated CCID register to control the context save operation when switching away from this context (as part of a subsequent MI_SET_CONTEXT command). This bit must be '1' when RS2 power state is enabled (via MCHBAR, offset 0x11B8)
	3	**Reserved** Project: Format:
	2	**Extended State Restore Enable** Project: {DevSNB] Format: U32 If set, the extended state identified in the Logical Context Data section of the Memory Data Formats chapter is loaded (or restored) as part of switching to this logical context. This method can be used to restore things such as filter coefficients using the indirect state restore followed by a restore of the extended logical context data. This bit affects the switch (if required) to the context specified in **Logical Context Address**. This bit will also be stored in the associated CCID register to control a subsequent context save operation when switching to this context (as part of a subsequent ring buffer switch). This bit must be '1' when RS2 power state is enabled (via MCHBAR, offset 0x11B8)
	2	**Reserved** Project: Format:

MI_SET_CONTEXT		
	1	**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.
	0	**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.

1.2.17 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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="5">0</td><td>31:29</td><td>Command Type
Default Value: 0h MI_COMMAND Format: OpCode</td></tr>
<tr><td>28:23</td><td>MI Command Opcode
Default Value: 20h MI_STORE_DATA_IMM Format: OpCode</td></tr>
<tr><td>22</td><td>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.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th><th>Project</th></tr>
<tr><td>0h</td><td>Reserved</td><td></td><td></td></tr>
<tr><td>1h</td><td>Global Graphics Address</td><td>This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.</td><td>All</td></tr>
</table></td></tr>
<tr><td>21:8</td><td>Reserved Project: All Format: MBZ</td></tr>
<tr><td>7:0</td><td>DWord Length
Default Value: 2h Excludes DWord (0,1) = 2 for DWord, 3 for QWord
Format: =n Total Length - 2</td></tr>
<tr><td>1</td><td>31:0</td><td>Reserved Project: All Format: MBZ</td></tr>
</table>

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	**Reserved** Project: All Format: MBZ
	0	Reserved Project: Format:
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).

1.2.18 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

MI_STORE_DATA_INDEX

	28:23	**MI Command Opcode** Default Value: 21h MI_STORE_DATA_INDEX Format: OpCode
	22	**Reserved** Project: Format:
	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 but must always be '0'
	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).

1.2.19 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

The following addresses should NOT be used for SRMs

1. 0x8800 - 0x88FF

2. >= 0x40000

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 24h MI_STORE_REGISTER_MEM Format: OpCode

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

Value	Name	Description	Project
0h	Reserved		
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

	21	**Reserved** Format: MBZ
	20:8	**Reserved** Project: All Format: MBZ
	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

MI_STORE_REGISTER_MEM		
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	**Reserved** Project: All Format: MBZ
	0	Reserved

1.2.20 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

1.2.21 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.

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 23h MI_UPDATE_GTT Format: OpCode
	22	**Use Global GTT** Project: All **Value** / **Name** / **Description** / **Project** 0h / Per Process Graphics Address / This command will use the Per Process GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer. / 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
	21:8	**Reserved** Project: All Format: MBZ

MI_UPDATE_GTT		
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
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.

1.2.22 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

1.2.23 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. Once parsed, the parser will halt (and suspend command arbitration) until the event/condition occurs. 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 CSunit is waiting for V-blank or flip done, HW can go into RC1/RC6 state.

Software must disable MI_WAIT_FOR_EVENT RC6 entry via RC_PSMI_CTRL if MI_WAIT_FOR_EVENT is parsed in a batch buffer with the following attributes set:

- batch buffer in PPGTT space (labeled “non-secure” in command)
- CB^2 batch buffer

MI_NOOP setting NOP register (or any other benign command) must be set after MI_WAIT_FOR_EVENT under the following conditions

- Back-to-back MI_WAIT_FOR_EVENT commands
- MI_WAIT_FOR_EVENT is the last command before head = tail

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 03h MI_WAIT_FOR_EVENT Format: OpCode
	22:20	**Reserved** Project: Format: MBZ
	22	**Reserved** Project: Format:
	21	**Reserved** Project: Format:

MI_WAIT_FOR_EVENT

	20	**Reserved** Project: Format:
	19:16	**Condition Code Wait Select** Project: All This field enables a wait for the duration that the corresponding condition code is active. These enable select one of 15 condition codes in the EXCC register, that cause the parser to wait until that condition-code in the EXCC is cleared.

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

Programming Notes
Note that not all condition codes are implemented. The parser operation is UNDEFINED if an unimplemented condition code is selected by this field.

	15:14	**Reserved** Project: Format: MBZ
	13	**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*.
	12	**Reserved** Project: All Format: MBZ
	11	**Display Pipe B Vertical Blank Wait Enable** Project: All Format: U32 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*).
	10	**Display Sprite B Flip Pending Wait Enable** Project: All 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).

MI_WAIT_FOR_EVENT		
	9	**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).
	8	**Display Pipe B Scan Line Wait Enable** Project: All Format: Enable This field enables a wait while a Display Pipe B “Scan Line” condition exists. This condition is defined as the the start of the scan line specified in the Pipe B Display Scan Line Count Range Compare Register.
	7:6	**Reserved** Project: All Format: MBZ
	5	**Display Pipe A H Blank Wait Enable** Project: All Format: U32 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.
	4	**Reserved** Project: All Format: MBZ
	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.
	2	**Display Sprite A Flip Pending Wait Enable** Project: All 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*.
	1	**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*.

MI_WAIT_FOR_EVENT		
	0	**Display Pipe A Scan Line Wait Enable** Project: All Format: Enable This field enables a wait while a Display Pipe A “Scan Line” condition exists. This condition is defined as the the start of the scan line specified in the Pipe A Display Scan Line Count Range Compare Register. See Scan Line Event in the Device Programming Interface chapter of *MI Functions*.

Revision History

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

§§