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

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

For the 2012 Intel® Core™ Processor Family

May 2012

Revision 1.0

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

Creative Commons License

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

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

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

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

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

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

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

Contents

1. Render Engine Command Streamer 4
1.1 Registers in Render Engine 4
1.1.1 Introduction 4
1.1.2 Outstanding Memory Requests Modulation Counters 6
1.1.3 Registers Used for Priority Field in Programmable Arbitration 9
1.1.4 Registers Used in Programmable Arbitration 12
1.1.5 Virtual Memory Control 29
1.1.6 GFX TLB In Use Virtual Address Registers 34
1.1.7 GFX Pending TLB Cycles Information Registers 39
1.1.8 Configuration Registers for Graphic Arbiter 45
1.1.9 Context Save Registers 47
1.1.10 Mode and Misc Ctrl Registers 54
1.1.11 RINGBUF — Ring Buffer Registers 75
1.1.12 Watchdog Timer Registers 81
1.1.13 Interrupt Control Registers 83
1.1.14 Logical Context Support 89
1.1.15 Pipelines Statistics Counter Registers 101
1.1.16 Predicate Render Registers 110
1.1.17 AUTO_DRAW Registers 112
1.1.18 MMIO Registers for GPGPU Indirect Dispatch 114
1.1.19 Memory Interface Registers 117
1.2 Memory Interface Commands for Rendering Engine 166
1.2.1 Introduction 166
1.2.2 Software Synchronization Commands 166
1.2.3 MI_ARB_CHECK 167
1.2.4 MI_ARB_ON_OFF 167
1.2.5 MI_BATCH_BUFFER_END 168
1.2.6 MI_CONDITIONAL_BATCH_BUFFER_END 168
1.2.7 MI_BATCH_BUFFER_START (Render) 169
1.2.8 MI_CLFLUSH 173
1.2.9 MI_DISPLAY_FLIP 174
1.2.10 MI_FLUSH 176
1.2.11 MI_LOAD_REGISTER_IMM 178
1.2.12 MI_NOOP 180
1.2.13 Surface Probing 180
1.2.14 MI_REPORT_HEAD 180
1.2.15 MI_SEMAPHORE_MBOX 181
1.2.16 MI_SET_CONTEXT 184
1.2.17 MI_STORE_DATA_IMM 186
1.2.18 MI_STORE_DATA_INDEX 187
1.2.19 MI_STORE_REGISTER_MEM 189
1.2.20 MI_SUSPEND_FLUSH 190
1.2.21 MI_UPDATE_GTT 191
1.2.22 MI_USER_INTERRUPT 192
1.2.23 MI_WAIT_FOR_EVENT 193
1.2.24 MI_LOAD_REGISTER_MEM 196
1.2.25 MI_URB_CLEAR 197
1.2.26 MI_PREDICATE 199
1.2.27 MI_TOPOLOGY_FILTER 201

1. Render Engine Command Streamer

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 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

ARB_MODE - Arbiter Mode Control register

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Size (in bits):	32
Trusted Type:	1
Address:	04030h

DWord	Bit	Description
0	31:16	**Mask Bits** Default Value: 0000000000000000b Access: RO Format: U16 Mask bits act as write enables for the bits in the lower bits of this register
	14	**GAM to Bypass GTT Translation (GAM2BGTTT)** Default Value: 0b Access: R/W Format: MBZ GAM to bypass GTT translation and pass logical addresses through with 0's padded on the MSBs to form the physical address.
	13	**DC GDR (DC_GDR)** Default Value: 0b Access: R/W DC GDR

ARB_MODE - Arbiter Mode Control register

12 **HIZ GDR (HIZ_GDR)**

Default Value:	0b
Access:	R/W

HIZ GDR

11 **STC GDR (STC_GDR)**

Default Value:	0b
Access:	R/W
Format:	U1

STC GDR

10 **BLB GDR (STC_GDR)**

Default Value:	0b
Access:	R/W

BLB GDR

9 **GAM PD GDR (GAMPD_GDR)**

Default Value:	0b
Access:	R/W

GAM PD GDR

8 **Color/Depth Port Share Bit (CDPS)**

Default Value:	00b
Access:	R/W
Format:	U1

Color/Depth port share bit

This bit is used to force Color and Depth Caches to share an arbiter read request port. By default (Bit = 0) the Color Cache will NOT share the read request port with the Depth Cache.

5 **Address Swizzling for Tiled Surfaces (AS4TS)**

Access:	R/W
Format:	U1

Address Swizzling for Tiled-Surfaces. 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
0b	No address Swizzling
1b	Address bit[6] needs to be swizzled for tiled surfaces

4 **VMC GDR Enable (VMC_GDR_EN)**

Access:	R/W

When this bit is set, Data requested from the VMC client will be generated by the GDR algorithm

3 **Texture Cache GDR Enable bit (TCGDREN)**

Access:	R/W
Format:	U1

Texture Cache GDR enable bit When this bit is set, Data requested from the Texture Cache client will be generated by the GDR algorithm (See GDR algorithm in xxx section)

ARB_MODE - Arbiter Mode Control register

	2	**Depth Cache GDR enable bit (DCGDREN)** Access: R/W Format: U1 When this bit is set, Data requested from the Depth Cache client will be generated by the GDR algorithm (See GDR algorithm in xxx section)	
	1	**Color Cache GDR enable bit(CCGDREN)** Access: R/W Format: U1 When this bit is set, Data requested from the Color Cache client will be generated by the GDR algorithm (See GDR algorithm in xxx section)	
	0	**GTT Accesses GDR (GTTAGDR)** Default Value: 0b Access: R/W Format: U1 When this bit is enabled along with the Client's GDR bit, PPGTT and GGTT requests for this memory access will also be tagged as GDR to SQ.	

1.1.2 Outstanding Memory Requests Modulation Counters

1.1.2.1 GFX_PEND_TLB_0 – Max Outstanding Pending TLB Requests 0

GFX_PEND_TLB_0 - Max Outstanding Pending TLB Requests 0

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	04034h-04037h

DWord	Bit	Description
0	31	**TEX Limit Enable bit** Format: U1 This bit is used to enable the pending TLB requests limitation function for the Texture Cache. When set, the number of internal pending read requests which require a TLB read will not exceed the programmed counter value.
	30	**Reserved** Format: MBZ

GFX_PEND_TLB_0 - Max Outstanding Pending TLB Requests 0

	29:24	**TEX TLB Limit Count**	Format:	U6	This is the MAX number of Allowed internal pending read requests which require a TLB read.
	23	**ISC Limit Enable bit**	Format:	U1	This bit is used to enable the pending TLB requests limitation function for the Instruction Cache. When set, the number of internal pending read requests which require a TLB read will not exceed the programmed counter value.
	22	**Reserved**	Format:	MBZ	
	21:16	**ISC TLB Limit Count**	Format:	U6	This is the MAX number of Allowed internal pending read requests which require a TLB read.
	15	**VF Limit Enable bit**	Format:	U1	This bit is used to enable the pending TLB requests limitation function for the Vertex Fetch. When set, the number of internal pending read requests which require a TLB read will not exceed the programmed counter value.
	14	**Reserved**	Format:	MBZ	
	13:8	**VF TLB Limit Count**	Format:	U6	This is the MAX number of Allowed internal pending read requests which require a TLB read.
	7	**CS Limit Enable bit**	Format:	U1	This bit is used to enable the pending TLB requests limitation function for the Command Streamer. When set, the number of internal pending read requests which require a TLB read will not exceed the programmed counter value.
	6	**Reserved**	Format:	MBZ	
	5:0	**CS TLB Limit Count**	Format:	U6	This is the MAX number of Allowed internal pending read requests which require a TLB read.

1.1.2.2 GFX_PEND_TLB_1 – Max Outstanding Pending TLB Requests 1

GFX_PEND_TLB_1 - Max Outstanding Pending TLB Requests 1

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	04038h-0403Bh

DWord	Bit	Description
0	31:16	**Reserved** Format: MBZ
	15	**RCZ Limit Enable bit** Format: U1 This bit is used to enable the pending TLB requests limitation function for the Render Depth Cache. When set, the number of internal pending read requests which require a TLB read will not exceed the programmed counter value.
	14	**Reserved** Format: MBZ
	13:8	**RCZ TLB Limit Count** Format: U6 This bit is used to enable the pending TLB requests limitation function for the Render Color Cache. When set, the number of internal pending read requests which require a TLB read will not exceed the programmed counter value.
	7	**RCC Limit Enable bit** Format: U1 This bit is used to enable the pending TLB requests limitation function for the Render Color Cache. When set, the number of internal pending read requests which require a TLB read will not exceed the programmed counter value.
	6	**Reserved** Format: MBZ
	5:0	**RCC TLB Limit Count** Format: U6 This is the MAX number of Allowed internal pending read requests which require a TLB read.

1.1.3 Registers Used for Priority Field in Programmable Arbitration

1.1.3.1 MIDARB_PRIO_HIT_REGISTER – Priority Field in Programmable Arbitration for Hit

MIDARB_PRIO_HIT_REGISTER - Priority Field in Programmable Arbitration for Hit

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	16
Trusted Type:	1
Address:	043A0h

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:12</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>11:9</td><td><b>Encoded Programmable Priority for MIDARB_GOTOFIELD_HIT3 Register</b>
<table>
<tr><th>Encoding</th><th>Priority 1</th><th>Priority 2</th><th>Priority 3</th></tr>
<tr><td>000</td><td>CS/VF/ISC</td><td>MT/CTC</td><td>RCC</td></tr>
<tr><td>001</td><td>CS/VF/ISC</td><td>RCC</td><td>MT/CTC</td></tr>
<tr><td>010</td><td>RCC</td><td>CS/VF/ISC</td><td>MT/CTC</td></tr>
<tr><td>011</td><td>RCC</td><td>MT/CTC</td><td>CS/VF/ISC</td></tr>
<tr><td>100</td><td>MT/CTC</td><td>CS/VF/ISC</td><td>RCC</td></tr>
<tr><td>101</td><td>MT/CTC</td><td>RCC</td><td>CS/VF/ISC</td></tr>
<tr><td>110</td><td>Reserved</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>111</td><td>Reserved</td><td>Reserved</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td></td><td>8:6</td><td><b>Encoded Programmable Priority for MIDARB_GOTOFIELD_HIT2 Register</b></td></tr>
<tr><td></td><td>5:3</td><td><b>Encoded Programmable Priority for MIDARB_GOTOFIELD_HIT1 Register</b></td></tr>
<tr><td></td><td>2:0</td><td><b>Encoded Programmable Priority for MIDARB_GOTOFIELD_HIT0 Register</b></td></tr>
</table>

1.1.3.2 MIDARB_PRIO_MISS_REGISTER – Priority Field in Programmable Arbitration for Miss

MIDARB_PRIO_MISS_REGISTER - Priority Field in Programmable Arbitration for Miss

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	04204h

DWord	Bit	Description
0	31:20	**Reserved**
	19:15	**Encoded Programmable Priority for MIDARB_GOTOFIELD_MISS3 Register**
	14:10	**Encoded Programmable Priority for MIDARB_GOTOFIELD_MISS2 Register**
	9:5	**Encoded Programmable Priority for MIDARB_GOTOFIELD_MISS1 Register**
	4:0	**Encoded Programmable Priority for MIDARB_GOTOFIELD_MISS0 Register**

1.1.3.3 MIDARB_PRIO_NP_REGISTER – Priority Field in Programmable Arbitration for Hit-NP

MIDARB_PRIO_NP_REGISTER - Priority Field in Programmable Arbitration for Hit-NP

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	043A4h
Address:	04208h

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:20</td><td><b>Reserved</b></td></tr>
<tr><td></td><td>19:15</td><td><b>Encoded Programmable Priority for MIDARB_GOTOFIELD_MISS3 Register</b>
<table>
<tr><th>Encoding</th><th>Priority 1</th><th>Priority 2</th><th>Priority 3</th><th>Priority 4</th></tr>
<tr><td>00000</td><td>CS/VF/ISC</td><td>MT_CTC</td><td>RCC</td><td>RCZ_HiZ_Stnc</td></tr>
</table>
</td></tr>
</table>

MIDARB_PRIO_NP_REGISTER - Priority Field in Programmable Arbitration for Hit-NP

00001	CS/VF/ISC	RCC	MT_CTC	RCZ_HiZ_Stnc
00010	RCC	CS/VF/ISC	MT_CTC	RCZ_HiZ_Stnc
00011	RCC	MT_CTC	CS/VF/ISC	RCZ_HiZ_Stnc
00100	MT_CTC	CS/VF/ISC	RCC	RCZ_HiZ_Stnc
00101	MT_CTC	RCC	CS/VF/ISC	RCZ_HiZ_Stnc
01000	CS/VF/ISC	MT_CTC	RCZ_HiZ_Stnc	RCC
01001	CS/VF/ISC	RCC	RCZ_HiZ_Stnc	MT_CTC
01010	RCC	CS/VF/ISC	RCZ_HiZ_Stnc	MT_CTC
01011	RCC	MT_CTC	RCZ_HiZ_Stnc	CS/VF/ISC
01100	MT_CTC	CS/VF/ISC	RCZ_HiZ_Stnc	RCC
01101	MT_CTC	RCC	RCZ_HiZ_Stnc	CS/VF/ISC
10000	CS/VF/ISC	RCZ_HiZ_Stnc	MT_CTC	RCC
10001	CS/VF/ISC	RCZ_HiZ_Stnc	RCC	MT_CTC
10010	RCC	RCZ_HiZ_Stnc	CS/VF/ISC	MT_CTC
10011	RCC	RCZ_HiZ_Stnc	MT_CTC	CS/VF/ISC
10100	MT_CTC	RCZ_HiZ_Stnc	CS/VF/ISC	RCC
10101	MT_CTC	RCZ_HiZ_Stnc	RCC	CS/VF/ISC
11000	RCZ_HiZ_Stnc	CS/VF/ISC	MT_CTC	RCC
11001	RCZ_HiZ_Stnc	CS/VF/ISC	RCC	MT_CTC
11010	RCZ_HiZ_Stnc	RCC	CS/VF/ISC	MT_CTC
11011	RCZ_HiZ_Stnc	RCC	MT_CTC	CS/VF/ISC
11100	RCZ_HiZ_Stnc	MT_CTC	CS/VF/ISC	RCC
11101	RCZ_HiZ_Stnc	MT_CTC	RCC	CS/VF/ISC
Other values	Reserved			

14:10	**Encoded Programmable Priority for MIDARB_GOTOFIELD_NP2 Register**
9:5	**Encoded Programmable Priority for MIDARB_GOTOFIELD_NP1 Register**
4:0	**Encoded Programmable Priority for MIDARB_GOTOFIELD_NP0 Register**

1.1.4 Registers Used in Programmable Arbitration

1.1.4.1 MIDARB_GOTOFIELD_HIT0_REGISTER – Goto Field in Programmable Arbitration for Hit0

MIDARB_GOTOFIELD_HIT0 - Goto Field in Programmable Arbitration for Hit0

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	16
Trusted Type:	1
Address:	043B0h

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:16</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>15:14</td><td><b>Goto field when request vector is 111</b>
Determines the GOTO and priority register to be used next:<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>00b</td><td></td><td>Use MIDARB_GOTOFIELD_HIT0 and MIDARB_PRIO_HIT_REGISTER[2:0]</td></tr><tr><td>01b</td><td></td><td>Use MIDARB_GOTOFIELD_HIT1 and MIDARB_PRIO_HIT_REGISTER[5:3]</td></tr><tr><td>10b</td><td></td><td>Use MIDARB_GOTOFIELD_HIT2 and MIDARB_PRIO_HIT_REGISTER[8:6]</td></tr><tr><td>11b</td><td></td><td>Use MIDARB_GOTOFIELD_HIT3 and MIDARB_PRIO_HIT_REGISTER[11:9]</td></tr></table></td></tr>
<tr><td></td><td>13:12</td><td><b>Goto field when request vector is 110b.</b></td></tr>
<tr><td></td><td>11:10</td><td><b>Goto field when request vector is 101b.</b></td></tr>
<tr><td></td><td>9:8</td><td><b>Goto field when request vector is 100b.</b></td></tr>
<tr><td></td><td>7:6</td><td><b>Goto field when request vector is 011b.</b></td></tr>
<tr><td></td><td>5:4</td><td><b>Goto field when request vector is 010b.</b></td></tr>
<tr><td></td><td>3:2</td><td><b>Goto field when request vector is 001b.</b></td></tr>
<tr><td></td><td>1:0</td><td><b>Goto field when request vector is 000b.</b></td></tr>
</table>

1.1.4.2 MIDARB_GOTOFIELD_HIT1_REGISTER – Goto Field in Programmable Arbitration for Hit1

<table>
<tr><th colspan="3">MIDARB_GOTOFIELD_HIT1 - Goto Field in Programmable Arbitration for Hit1</th></tr>
<tr><td colspan="2">Register Space:</td><td>MMIO: 0/2/0</td></tr>
<tr><td colspan="2">Source:</td><td>RenderCS</td></tr>
<tr><td colspan="2">Default Value:</td><td>0x00000000</td></tr>
<tr><td colspan="2">Access:</td><td>R/W</td></tr>
<tr><td colspan="2">Size (in bits):</td><td>16</td></tr>
<tr><td colspan="2">Trusted Type:</td><td>1</td></tr>
<tr><td colspan="2">Address:</td><td>043B4h</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="9">0</td><td>31:16</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>15:14</td><td><b>Goto field when request vector is 111</b>
Determines the GOTO and priority register to be used next<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>00b</td><td></td><td>Use MIDARB_GOTOFIELD_HIT0 and MIDARB_PRIO_HIT_REGISTER[2:0]</td></tr>
<tr><td>01b</td><td></td><td>Use MIDARB_GOTOFIELD_HIT1 and MIDARB_PRIO_HIT_REGISTER[5:3]</td></tr>
<tr><td>10b</td><td></td><td>Use MIDARB_GOTOFIELD_HIT2 and MIDARB_PRIO_HIT_REGISTER[8:6]</td></tr>
<tr><td>11b</td><td></td><td>Use MIDARB_GOTOFIELD_HIT3 and MIDARB_PRIO_HIT_REGISTER[11:9]</td></tr>
</table></td></tr>
<tr><td>13:12</td><td><b>Goto field when request vector is 110b.</b></td></tr>
<tr><td>11:10</td><td><b>Goto field when request vector is 101b.</b></td></tr>
<tr><td>9:8</td><td><b>Goto field when request vector is 100b.</b></td></tr>
<tr><td>7:6</td><td><b>Goto field when request vector is 011b.</b></td></tr>
<tr><td>5:4</td><td><b>Goto field when request vector is 010b.</b></td></tr>
<tr><td>3:2</td><td><b>Goto field when request vector is 001b.</b></td></tr>
<tr><td>1:0</td><td><b>Goto field when request vector is 000b.</b></td></tr>
</table>

1.1.4.3 MIDARB_GOTOFIELD_HIT2_REGISTER – Goto Field in Programmable Arbitration for Hit2

<table>
<tr><th colspan="2">MIDARB_GOTOFIELD_HIT2 - Goto Field in Programmable Arbitration for Hit2</th></tr>
<tr><td>Register Space:</td><td>MMIO: 0/2/0</td></tr>
<tr><td>Source:</td><td>RenderCS</td></tr>
<tr><td>Default Value:</td><td>0x00000000</td></tr>
<tr><td>Access:</td><td>R/W</td></tr>
<tr><td>Size (in bits):</td><td>16</td></tr>
<tr><td>Trusted Type:</td><td>1</td></tr>
<tr><td>Address:</td><td>043B8h</td></tr>
</table>

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:16</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>15:14</td><td><b>Goto field when request vector is 111.</b>
Determines the GOTO and priority register to be used next<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>00b</td><td></td><td>Use MIDARB_GOTOFIELD_HIT0 and MIDARB_PRIO_HIT_REGISTER[2:0]</td></tr>
<tr><td>01b</td><td></td><td>Use MIDARB_GOTOFIELD_HIT1 and MIDARB_PRIO_HIT_REGISTER[5:3]</td></tr>
<tr><td>10b</td><td></td><td>Use MIDARB_GOTOFIELD_HIT2 and MIDARB_PRIO_HIT_REGISTER[8:6]</td></tr>
<tr><td>11b</td><td></td><td>Use MIDARB_GOTOFIELD_HIT3 and MIDARB_PRIO_HIT_REGISTER[11:9]</td></tr>
</table></td></tr>
<tr><td></td><td>13:12</td><td><b>Goto field when request vector is 110b.</b></td></tr>
<tr><td></td><td>11:10</td><td><b>Goto field when request vector is 101b.</b></td></tr>
<tr><td></td><td>9:8</td><td><b>Goto field when request vector is 100b.</b></td></tr>
<tr><td></td><td>7:6</td><td><b>Goto field when request vector is 011b.</b></td></tr>
<tr><td></td><td>5:4</td><td><b>Goto field when request vector is 010b.</b></td></tr>
<tr><td></td><td>3:2</td><td><b>Goto field when request vector is 001b.</b></td></tr>
<tr><td></td><td>1:0</td><td><b>Goto field when request vector is 000b.</b></td></tr>
</table>

1.1.4.4 MIDARB_GOTOFIELD_HIT3_REGISTER – Goto Field in Programmable Arbitration for Hit3

MIDARB_GOTOFIELD_HIT3 - Goto Field in Programmable Arbitration for Hit3

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	16
Trusted Type:	1
Address:	043BCh

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:16</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>15:14</td><td><b>Goto field when request vector is 111.</b>
Determines the GOTO and priority register to be used next.
Field for arbitration on next clock cycle for request entries of 111 corresponding to arbitration action field entry of MIDARB_PRIO_HIT_REGISTER[11:9]<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>00b</td><td></td><td>Use MIDARB_GOTOFIELD_HIT0 and MIDARB_PRIO_HIT_REGISTER[2:0]</td></tr><tr><td>01b</td><td></td><td>Use MIDARB_GOTOFIELD_HIT1 and MIDARB_PRIO_HIT_REGISTER[5:3]</td></tr><tr><td>10b</td><td></td><td>Use MIDARB_GOTOFIELD_HIT2 and MIDARB_PRIO_HIT_REGISTER[8:6]</td></tr><tr><td>11b</td><td></td><td>Use MIDARB_GOTOFIELD_HIT3 and MIDARB_PRIO_HIT_REGISTER[11:9]</td></tr></table></td></tr>
<tr><td></td><td>13:12</td><td><b>Goto field when request vector is 110.</b>
Field for arbitration on next clock cycle for request entries of 110 corresponding to arbitration action field entry of MIDARB_PRIO_HIT_REGISTER[11:9]</td></tr>
<tr><td></td><td>11:10</td><td><b>Goto field when request vector is 101.</b>
Field for arbitration on next clock cycle for request entries of 101 corresponding to arbitration action field entry of MIDARB_PRIO_HIT_REGISTER[11:9]</td></tr>
<tr><td></td><td>9:8</td><td><b>Goto field when request vector is 100.</b>
Field for arbitration on next clock cycle for request entries of 100 corresponding to arbitration action field entry of MIDARB_PRIO_HIT_REGISTER[11:9]</td></tr>
<tr><td></td><td>7:6</td><td><b>Goto field when request vector is 011.</b>
Field for arbitration on next clock cycle for request entries of 011 corresponding to arbitration action field entry of MIDARB_PRIO_HIT_REGISTER[11:9]</td></tr>
<tr><td></td><td>5:4</td><td><b>Goto field when request vector is 010.</b>
Field for arbitration on next clock cycle for request entries of 010 corresponding to arbitration action field entry of MIDARB_PRIO_HIT_REGISTER[11:9]</td></tr>
<tr><td></td><td>3:2</td><td><b>Goto field when request vector is 001.</b>
Field for arbitration on next clock cycle for request entries of 001 corresponding to arbitration action field entry of MIDARB_PRIO_HIT_REGISTER[11:9]</td></tr>
<tr><td></td><td>1:0</td><td><b>Goto field when request vector is 000.</b>
Field for arbitration on next clock cycle for request entries of 000 corresponding to arbitration action field entry of MIDARB_PRIO_HIT_REGISTER[11:9]</td></tr>
</table>

1.1.4.5 MIDARB_GOTOFIELD_NP0_REGISTER – Goto Field in Programmable Arbitration for Hit-NP0

MIDARB_GOTOFIELD_NP0 - Goto Field in Programmable Arbitration for Hit-NP0

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	043C0h

DWord	Bit	Description
0	31:30	**Goto field when request vector is 1111.** Determines the GOTO and priority register to be used next. Value | Name | Description 00b | | Use MIDARB_GOTOFIELD_NP0 and MIDARB_PRIO_NP_REGISTER[4:0] 01b | | Use MIDARB_GOTOFIELD_NP1 and MIDARB_PRIO_NP_REGISTER[9:5] 10b | | Use MIDARB_GOTOFIELD_NP2 and MIDARB_PRIO_NP_REGISTER[14:10] 11b | | Use MIDARB_GOTOFIELD_NP3 and MIDARB_PRIO_NP_REGISTER[19:15]
	29:28	**Goto field when request vector is 1110b.**
	27:26	**Goto field when request vector is 1101b.**
	25:24	**Goto field when request vector is 1100b.**
	23:22	**Goto field when request vector is 1011b.**
	21:20	**Goto field when request vector is 1010b.**
	19:18	**Goto field when request vector is 1001b.**
	17:16	**Goto field when request vector is 1000b.**
	15:14	**Goto field when request vector is 0111b.**
	13:12	**Goto field when request vector is 0110b.**
	11:10	**Goto field when request vector is 0101b.**
	9:8	**Goto field when request vector is 0100b.**
	7:6	**Goto field when request vector is 0011b.**
	5:4	**Goto field when request vector is 0010b.**
	3:2	**Goto field when request vector is 0001b.**
	1:0	**Goto field when request vector is 0000b.**

1.1.4.6 MIDARB_GOTOFIELD_NP1_REGISTER – Goto Field in Programmable Arbitration for Hit-NP1

MIDARB_GOTOFIELD_NP1 - Goto Field in Programmable Arbitration for Hit-NP1

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	043C4h

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:30</td><td><b>Goto field when request vector is 1111.</b>
Determines the GOTO and priority register to be used next.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>00b</td><td></td><td>Use MIDARB_GOTOFIELD_NP0 and MIDARB_PRIO_NP_REGISTER[4:0]</td></tr>
<tr><td>01b</td><td></td><td>Use MIDARB_GOTOFIELD_NP1 and MIDARB_PRIO_NP_REGISTER[9:5]</td></tr>
<tr><td>10b</td><td></td><td>Use MIDARB_GOTOFIELD_NP2 and MIDARB_PRIO_NP_REGISTER[14:10]</td></tr>
<tr><td>11b</td><td></td><td>Use MIDARB_GOTOFIELD_NP3 and MIDARB_PRIO_NP_REGISTER[19:15]</td></tr>
</table></td></tr>
<tr><td></td><td>29:28</td><td><b>Goto field when request vector is 1110b.</b></td></tr>
<tr><td></td><td>27:26</td><td><b>Goto field when request vector is 1101b.</b></td></tr>
<tr><td></td><td>25:24</td><td><b>Goto field when request vector is 1100b.</b></td></tr>
<tr><td></td><td>23:22</td><td><b>Goto field when request vector is 1011b.</b></td></tr>
<tr><td></td><td>21:20</td><td><b>Goto field when request vector is 1010b.</b></td></tr>
<tr><td></td><td>19:18</td><td><b>Goto field when request vector is 1001b.</b></td></tr>
<tr><td></td><td>17:16</td><td><b>Goto field when request vector is 1000b.</b></td></tr>
<tr><td></td><td>15:14</td><td><b>Goto field when request vector is 0111b.</b></td></tr>
<tr><td></td><td>13:12</td><td><b>Goto field when request vector is 0110b.</b></td></tr>
<tr><td></td><td>11:10</td><td><b>Goto field when request vector is 0101b.</b></td></tr>
<tr><td></td><td>9:8</td><td><b>Goto field when request vector is 0100b.</b></td></tr>
<tr><td></td><td>7:6</td><td><b>Goto field when request vector is 0011b.</b></td></tr>
<tr><td></td><td>5:4</td><td><b>Goto field when request vector is 0010b.</b></td></tr>
<tr><td></td><td>3:2</td><td><b>Goto field when request vector is 0001b.</b></td></tr>
<tr><td></td><td>1:0</td><td><b>Goto field when request vector is 0000b.</b></td></tr>
</table>

1.1.4.7 MIDARB_GOTOFIELD_NP2_REGISTER – Goto Field in Programmable Arbitration for Hit-NP2

MIDARB_GOTOFIELD_NP2 - Goto Field in Programmable Arbitration for Hit-NP2

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	043C8h

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:30</td><td><b>Goto field when request vector is 1111.</b>
Determines the GOTO and priority register to be used next.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>00b</td><td></td><td>Use MIDARB_GOTOFIELD_NP0 and MIDARB_PRIO_NP_REGISTER[4:0]</td></tr>
<tr><td>01b</td><td></td><td>Use MIDARB_GOTOFIELD_NP1 and MIDARB_PRIO_NP_REGISTER[9:5]</td></tr>
<tr><td>10b</td><td></td><td>Use MIDARB_GOTOFIELD_NP2 and MIDARB_PRIO_NP_REGISTER[14:10]</td></tr>
<tr><td>11b</td><td></td><td>Use MIDARB_GOTOFIELD_NP3 and MIDARB_PRIO_NP_REGISTER[19:15]</td></tr>
</table></td></tr>
<tr><td></td><td>29:28</td><td><b>Goto field when request vector is 1110b.</b></td></tr>
<tr><td></td><td>27:26</td><td><b>Goto field when request vector is 1101b.</b></td></tr>
<tr><td></td><td>25:24</td><td><b>Goto field when request vector is 1100b.</b></td></tr>
<tr><td></td><td>23:22</td><td><b>Goto field when request vector is 1011b.</b></td></tr>
<tr><td></td><td>21:20</td><td><b>Goto field when request vector is 1010b.</b></td></tr>
<tr><td></td><td>19:18</td><td><b>Goto field when request vector is 1001b.</b></td></tr>
<tr><td></td><td>17:16</td><td><b>Goto field when request vector is 1000b.</b></td></tr>
<tr><td></td><td>15:14</td><td><b>Goto field when request vector is 0111b.</b></td></tr>
<tr><td></td><td>13:12</td><td><b>Goto field when request vector is 0110b.</b></td></tr>
<tr><td></td><td>11:10</td><td><b>Goto field when request vector is 0101b.</b></td></tr>
<tr><td></td><td>9:8</td><td><b>Goto field when request vector is 0100b.</b></td></tr>
<tr><td></td><td>7:6</td><td><b>Goto field when request vector is 0011b.</b></td></tr>
<tr><td></td><td>5:4</td><td><b>Goto field when request vector is 0010b.</b></td></tr>
<tr><td></td><td>3:2</td><td><b>Goto field when request vector is 0001b.</b></td></tr>
<tr><td></td><td>1:0</td><td><b>Goto field when request vector is 0000b.</b></td></tr>
</table>

1.1.4.8 MIDARB_GOTOFIELD_NP3_REGISTER – Goto Field in Programmable Arbitration for Hit-NP3

MIDARB_GOTOFIELD_NP3 - Goto Field in Programmable Arbitration for Hit-NP3

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	043CCh

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:30</td><td><b>Goto field when request vector is 1111.</b>
Determines the GOTO and priority register to be used next.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>00b</td><td></td><td>Use MIDARB_GOTOFIELD_NP0 and MIDARB_PRIO_NP_REGISTER[4:0]</td></tr>
<tr><td>01b</td><td></td><td>Use MIDARB_GOTOFIELD_NP1 and MIDARB_PRIO_NP_REGISTER[9:5]</td></tr>
<tr><td>10b</td><td></td><td>Use MIDARB_GOTOFIELD_NP2 and MIDARB_PRIO_NP_REGISTER[14:10]</td></tr>
<tr><td>11b</td><td></td><td>Use MIDARB_GOTOFIELD_NP3 and MIDARB_PRIO_NP_REGISTER[19:15]</td></tr>
</table></td></tr>
<tr><td></td><td>29:28</td><td><b>Goto field when request vector is 1110b.</b></td></tr>
<tr><td></td><td>27:26</td><td><b>Goto field when request vector is 1101b.</b></td></tr>
<tr><td></td><td>25:24</td><td><b>Goto field when request vector is 1100b.</b></td></tr>
<tr><td></td><td>23:22</td><td><b>Goto field when request vector is 1011b.</b></td></tr>
<tr><td></td><td>21:20</td><td><b>Goto field when request vector is 1010b.</b></td></tr>
<tr><td></td><td>19:18</td><td><b>Goto field when request vector is 1001b.</b></td></tr>
<tr><td></td><td>17:16</td><td><b>Goto field when request vector is 1000b.</b></td></tr>
<tr><td></td><td>15:14</td><td><b>Goto field when request vector is 0111b.</b></td></tr>
<tr><td></td><td>13:12</td><td><b>Goto field when request vector is 0110b.</b></td></tr>
<tr><td></td><td>11:10</td><td><b>Goto field when request vector is 0101b.</b></td></tr>
<tr><td></td><td>9:8</td><td><b>Goto field when request vector is 0100b.</b></td></tr>
<tr><td></td><td>7:6</td><td><b>Goto field when request vector is 0011b.</b></td></tr>
<tr><td></td><td>5:4</td><td><b>Goto field when request vector is 0010b.</b></td></tr>
<tr><td></td><td>3:2</td><td><b>Goto field when request vector is 0001b.</b></td></tr>
<tr><td></td><td>1:0</td><td><b>Goto field when request vector is 0000b.</b></td></tr>
</table>

1.1.4.9 ARB_GAC_GAM_REQCNTS0 – GAC_GAM Arbitration Counters Register 0

ARB_GAC_GAM_REQCNTS0 - GAC_GAM Arbitration Counters Register 0

Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	043A8h

DWord	Bit	Description
0	31:22	**Reserved**
	21:16	**Number of GAC WR requests to be accumulated before applying the arbitration**
	15:14	**Reserved**
	13:8	**Number of GAC R requests to be accumulated before applying the arbitration**
	7:6	**Reserved**
	5:0	**Number of GAC RO requests to be accumulated before applying the arbitration**

1.1.4.10 ARB_GAC_GAM_REQCNTS1 – GAC_GAM Arbitration Counters Register 1

ARB_GAC_GAM_REQCNTS1 - GAC_GAM Arbitration Counters Register 1

Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	043ACh

DWord	Bit	Description
0	31:22	**Reserved**
	21:16	**Number of GAC WR requests to be accumulated before applying the arbitration**
	15:14	**Reserved**
	13:8	**Number of GAC R requests to be accumulated before applying the arbitration**
	7:6	**Reserved**
	5:0	**Number of GAC RO requests to be accumulated before applying the arbitration**

1.1.4.11 ARB_RO_GAC_GAM0 – GAC_GAM RO Arbitration Register 0

ARB_RO_GAC_GAM0 - GAC_GAM RO Arbitration Register 0	
Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	043D0h

DWord	Bit	Description
0	31:28	**Reserved**
	27	**Priority for entry 1**
	26:24	**Goto field for entry 1 when request vector is 11b**
	23:21	**Goto field for entry 1 when request vector is 10b**
	20:18	**Goto field for entry 1 when request vector is 01b**
	17:15	**Goto field for entry 1 when request vector is 00b**
	14:13	**Reserved**
	12	**Priority for entry 01**
	11:9	**Goto field for entry 01 when request vector is 11b**
	8:6	**Goto field for entry 01 when request vector is 10b**
	5:3	**Goto field for entry 01 when request vector is 01b**
	2:0	**Goto field for entry 01 when request vector is 00b**

1.1.4.12 ARB_RO_GAC_GAM1 – GAC_GAM RO Arbitration Register 1

ARB_RO_GAC_GAM1 - GAC_GAM RO Arbitration Register 1	
Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	043D4h

DWord	Bit	Description
0	31:28	**Reserved**
	27	**Priority for entry 3**

ARB_RO_GAC_GAM1 - GAC_GAM RO Arbitration Register 1

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

1.1.4.13 ARB_RO_GAC_GAM2 – GAC_GAM RO Arbitration Register 2

ARB_RO_GAC_GAM2 - GAC_GAM RO Arbitration Register 2

Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	043D8h

DWord	Bit	Description
0	31:28	**Reserved**
	27	**Priority for entry 5**
	26:24	**Goto field for entry 5 when request vector is 11b**
	23:21	**Goto field for entry 5 when request vector is 10b**
	20:18	**Goto field for entry 5 when request vector is 01b**
	17:15	**Goto field for entry 5 when request vector is 00b**
	14:13	**Reserved**
	12	**Priority for entry 4**
	11:9	**Goto field for entry 4 when request vector is 11b**
	8:6	**Goto field for entry 4 when request vector is 10b**
	5:3	**Goto field for entry 4 when request vector is 01b**
	2:0	**Goto field for entry 4 when request vector is 00b**

1.1.4.14 ARB_RO_GAC_GAM3 – GAC_GAM RO Arbitration Register 3

ARB_RO_GAC_GAM3 - GAC_GAM RO Arbitration Register 3	
Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	043DCh

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

1.1.4.15 ARB_R_GAC_GAM0 – GAC_GAM R Arbitration Register 0

ARB_R_GAC_GAM0 - GAC_GAM R Arbitration Register 0	
Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	043E0h

DWord	Bit	Description
0	31:28	**Reserved**
	27	**Priority for entry 1**
	26:24	**Goto field for entry 1 when request vector is 11b**

ARB_R_GAC_GAM0 - GAC_GAM R Arbitration Register 0

	23:21	**Goto field for entry 1 when request vector is 10b**
	20:18	**Goto field for entry 1 when request vector is 01b**
	17:15	**Goto field for entry 1 when request vector is 00b**
	14:13	**Reserved**
	12	**Priority for entry 0**
	11:9	**Goto field for entry 0 when request vector is 11b**
	8:6	**Goto field for entry 0 when request vector is 10b**
	5:3	**Goto field for entry 0 when request vector is 01b**
	2:0	**Goto field for entry 0 when request vector is 00b**

1.1.4.16 ARB_R_GAC_GAM1 – GAC_GAM R Arbitration Register 1

ARB_R_GAC_GAM1 - GAC_GAM R Arbitration Register 1

Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	043E4h

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

1.1.4.17 ARB_R_GAC_GAM2 – GAC_GAM R Arbitration Register 2

ARB_R_GAC_GAM2 - GAC_GAM R Arbitration Register 2	
Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	043E8h

DWord	Bit	Description
0	31:28	**Reserved**
	27	**Priority for entry 5**
	26:24	**Goto field for entry 5 when request vector is 11b**
	23:21	**Goto field for entry 5 when request vector is 10b**
	20:18	**Goto field for entry 5 when request vector is 01b**
	17:15	**Goto field for entry 5 when request vector is 00b**
	14:13	**Reserved**
	12	**Priority for entry 4**
	11:9	**Goto field for entry 4 when request vector is 11b**
	8:6	**Goto field for entry 4 when request vector is 10b**
	5:3	**Goto field for entry 4 when request vector is 01b**
	2:0	**Goto field for entry 4 when request vector is 00b**

1.1.4.18 ARB_R_GAC_GAM3 – GAC_GAM R Arbitration Register 3

ARB_R_GAC_GAM3 - GAC_GAM R Arbitration Register 3	
Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	043ECh

DWord	Bit	Description
0	31:28	**Reserved**
	27	**Priority for entry 7**
	26:24	**Goto field for entry 7 when request vector is 11b**

ARB_R_GAC_GAM3 - GAC_GAM R Arbitration Register 3

	23:21	**Goto field for entry 7 when request vector is 10b**
	20:18	**Goto field for entry 7 when request vector is 01b**
	17:15	**Goto field for entry 7 when request vector is 00b**
	14:13	**Reserved**
	12	**Priority for entry 6**
	11:9	**Goto field for entry 6 when request vector is 11b**
	8:6	**Goto field for entry 6 when request vector is 10b**
	5:3	**Goto field for entry 6 when request vector is 01b**
	2:0	**Goto field for entry 6 when request vector is 00b**

1.1.4.19 ARB_WR_GAC_GAM0 – GAC_GAM WR Arbitration Register 0

ARB_WR_GAC_GAM0 - GAC_GAM WR Arbitration Register 0

Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	043F0h

DWord	Bit	Description
0	31:28	**Reserved**
	27	**Priority for entry 1**
	26:24	**Goto field for entry 1 when request vector is 11b**
	23:21	**Goto field for entry 1 when request vector is 10b**
	20:18	**Goto field for entry 1 when request vector is 01b**
	17:15	**Goto field for entry 1 when request vector is 00b**
	14:13	**Reserved**
	12	**Priority for entry 0**
	11:9	**Goto field for entry 0 when request vector is 11b**
	8:6	**Goto field for entry 0 when request vector is 10b**
	5:3	**Goto field for entry 0 when request vector is 01b**
	2:0	**Goto field for entry 0 when request vector is 00b**

1.1.4.20 ARB_WR_GAC_GAM1 – GAC_GAM WR Arbitration Register 1

ARB_WR_GAC_GAM1 - GAC_GAM WR Arbitration Register 1	
Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	043F4h

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

1.1.4.21 ARB_WR_GAC_GAM2 – GAC_GAM WR Arbitration Register 2

ARB_WR_GAC_GAM2 - GAC_GAM WR Arbitration Register 2	
Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	043F8h

DWord	Bit	Description
0	31:28	**Reserved**
	27	**Priority for entry 5**
	26:24	**Goto field for entry 5 when request vector is 11b**

ARB_WR_GAC_GAM2 - GAC_GAM WR Arbitration Register 2

	23:21	**Goto field for entry 5 when request vector is 10b**
	20:18	**Goto field for entry 5 when request vector is 01b**
	17:15	**Goto field for entry 5 when request vector is 00b**
	14:13	**Reserved**
	12	**Priority for entry 4**
	11:9	**Goto field for entry 4 when request vector is 11b**
	8:6	**Goto field for entry 4 when request vector is 10b**
	5:3	**Goto field for entry 4 when request vector is 01b**
	2:0	**Goto field for entry 4 when request vector is 00b**

1.1.4.22 ARB_WR_GAC_GAM3 – GAC_GAM WR Arbitration Register 3

ARB_WR_GAC_GAM3 - GAC_GAM WR Arbitration Register 3

Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	043FCh

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

1.1.5 Virtual Memory Control

1.1.5.1 HWS_PGA — Hardware Status Page Address Register

Programming Note: If this register is written, a workload must subsequently be dispatched to the render command streamer.

HWS_PGA - Hardware Status Page Address Register

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	04080h

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

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

DWord	Bit	Description
0	31:12	**Address** Format: GraphicsAddress[31:12] This field is used by SW to specify Bits 31:12 of the 4 KB-aligned System Memory address of the 4 KB page known as the Hardware Status Page. The Global GTT is used to map this page from the graphics virtual address to physical address. **Programming Notes** If the Per-Process Virtual Address Space and Exec List Enable bit is set, HW requires that the status page is programmed to allow for the context switch status to be reported.
	11:0	**Reserved** Format: MBZ

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

Hardware Status Page Layout

Source:	RenderCS
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000. 0x00000000. 0x00000000. 0x00000000. 0x00000000. 0x00000000. 0x00000000.

Hardware Status Page Layout

```
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
```

Hardware Status Page Layout

```
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000. 0x00000000. 0x00000000. 0x00000000. 0x00000000. 0x00000000. 0x00000000.
```

Hardware Status Page Layout

```
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000
```

DWord	Bit	Description
0	31:0	**Interrupt Status Register Storage** Project: All 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.
1..3	31:0	**Reserved** Project: All Must not be used.
4	31:0	**Ring Head Pointer Storage** Project: All 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).
5..15	31:0	**Reserved** Project: All

Hardware Status Page Layout

		Must not be used.
16..27	31:0	**Context Status DWords** Project: All
28..30	31:0	**Reserved** Project: All Must not be used.
31	31:0	**Last Written Status Offset** Project: All
32..1023	31:0	**General Purpose** Project: All These locations can be used for general purpose via the MI_STORE_DATA_INDEX or MI_STORE_DATA_IMM instructions.

1.1.5.2 PP_DCLV – PPGTT Directory Cacheline Valid Register

PP_DCLV - PPGTT Directory Cacheline Valid Register

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000, 0x00000000
Size (in bits):	64
Address:	02220h

Description	Project
Access: R/W	
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 process's 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.	

Programming Notes	Project
Page Directory Base Register is a Global Context Register (power context) and not maintained per context in ring buffer mode of submission. One should explicitly load PP_DCLV followed by PP_DIR_BASE register through Load Register Immediate commands in Ring Buffer before submitting a context. One should program these registers after ensuring the pipe is completely flushed with TLB’s invalidated.	

DWord	Bit	Description
0	63:32	**Reserved** Project: All

PP_DCLV - PPGTT Directory Cacheline Valid Register

		Format:	MBZ
	31:0	**PPGTT Directory Cache Restore [1..32] 16 entries** Project: All Format: BitMask[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.	

1.1.6 GFX TLB In Use Virtual Address Registers

1.1.6.1 MTTLB_VA — MT Virtual Page Address Registers

MTTLB_VA - MT Virtual Page Address Registers

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	04800h-048FCh

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

1.1.6.2 MTTLB_VLD — Valid Bit Vector 0 for MTTLB

MTTLB_VLD0 - Valid Bit Vector 0 for MTTLB	
Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	04780h-04783h

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

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

1.1.6.3 MTTLB_VLD — Valid Bit Vector 1 for MTTLB

MTTLB_VLD1 - Valid Bit Vector 1 for MTTLB	
Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	04784h-04787h

This register contains the valid bits for entries 0-31 of MTTLB (Texture and constant cache TLBVertex Fetch, Instruction Cache, and Command Streamer TLB).

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

1.1.6.4 VICTLB_VA — VIC Virtual page Address Registers

VICTLB_VA - VIC Virtual page Address Registers	
Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	04900h-049FCh

These registers are directly mapped to the current Virtual Addresses in the VICTLB (Vertex Fetch, Instruction Cache, and Command Streamer TLB.)

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

1.1.6.5 VICTLB_VLD — Valid Bit Vector 0 for MTVICTLB

VICTLB_VLD0 - Valid Bit Vector 0 for MTVICTLB	
Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	04788h-0478Bh

This register contains the valid bits for entries 0-31 of VICTLB (Vertex Fetch, Instruction Cache, and Command Streamer TLB).

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

1.1.6.6 VICTLB_VLD — Valid Bit Vector 1 for MTVICTLB

MTVICTLB_VLD1 - Valid Bit Vector 1 for MTVICTLB	
Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	0478Ch-0478Fh

This register contains the valid bits for entries 0-31 of VICTLB (Vertex Fetch, Instruction Cache, and Command Streamer TLB).

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

1.1.6.7 RCCTLB_VA — Virtual page Address Registers

RCCTLB_VA - RCC Virtual page Address Registers	
Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	04A00h-04AFCh

These registers are directly mapped to the current Virtual Addresses in the RCCTLB (Render Cache for Color TLB).

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

1.1.6.8 RCCTLB_VLD — Valid Bit Vector 0 for RCCTLB

RCCTLB_VLD0 - Valid Bit Vector 0 for RCCTLB

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	04790h-04793h

This register contains the valid bits for entries 0-31 of RCCTLB (Render Cache for Color TLB).

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

1.1.6.9 RCZTLB_VA — RCZ Virtual Page Address Registers

RCZTLB_VA - RCZ Virtual Page Address Registers

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	04B00h-04BFCh

These registers are directly mapped to the current Virtual Addresses in the RCZTLB (Render Cache for Z (Depth), Hi Z, and Stencil TLB).

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

1.1.6.10 RCZTLB_VLD0 — Valid Bit Vector 0 for RCZTLB

RCZTLB_VLD0 - Valid Bit Vector 0 for RCZTLB	
Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	04798h-0479Bh

This register contains the valid bits for entries 0-31 of RCZTLB (Render Cache for Z (Depth), Hi Z, and Stencil TLB).

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

1.1.6.11 RCZTLB_VLD1 — Valid Bit Vector 1 for RCZTLB

RCZTLB_VLD1 - Valid Bit Vector 1 for RCZTLB	
Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Trusted Type:	1
Address:	0479Ch-0479Fh

This register contains the valid bits for entries 0-31 of RCZTLB (Render Cache for Z (Depth), Hi Z, and Stencil TLB).

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

1.1.7 GFX Pending TLB Cycles Information Registers

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

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

1.1.7.1 TLBPEND_VLD0 - Valid Bit Vector 0 for TLBPEND Registers

TLBPEND_VLD0 - Valid Bit Vector 0 for TLBPEND registers

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	04700h-04703h

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

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

1.1.7.2 TLBPEND_VLD1 - Valid Bit Vector 1 for TLBPEND Registers

TLBPEND_VLD1 - Valid Bit Vector 1 for TLBPEND registers

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	04704h-04707h

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

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

1.1.7.3 TLBPEND_RDY0 - Ready Bit Vector 0 for TLBPEND Registers

TLBPEND_RDY0 - Ready Bit Vector 0 for TLBPEND registers	
Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	04708h-0470Bh

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

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

1.1.7.4 TLBPEND_RDY1 - Ready Bit Vector 1 for TLBPEND Registers

TLBPEND_RDY1 - Ready Bit Vector 1 for TLBPEND registers	
Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	0470Ch-0470Fh

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

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

1.1.7.5 TLBPEND_SEC0 — Section 0 of TLBPEND Entry

TLBPEND_SEC0 - Section 0 of TLBPEND Entry	
Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	04400h-044FCh
This register is directly mapped to the TLBPEND Array in the Graphic Arbiter.	

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

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

1.1.7.6 TLBPEND_SEC1 — Section 1 of TLBPEND entry

TLBPEND_SEC1 - Section 1 of TLBPEND Entry

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	04500h-045FCh

This register is directly mapped to the current Virtual Addresses in the MTTLB (Texture and constant cache TLBRender Cache for Z (Depth), Hi Z, and Stencil TLB).

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:28</td><td><b>Current address</b>
Bits 9:6 of the Virtual Address of the cycle.</td></tr>
<tr><td></td><td>27:24</td><td><b>Cacheability Control Bits</b>
Bits 3:1 of the GTT entry used to translate the Virtual Address. 000 if translation is pending.
3
2 <b>Graphics Data Type (GFDT).</b> This field contains the GFDT bit for this surface when writes occur. GFDT can also be set by the GTT. The effective GFDT is the logical OR of this field with the GFDT from the GTT entry. This field is ignored for reads.
1:0 <b>Cacheability Control.</b> This field controls cacheability in the mid-level cache (MLC) and last-level cache (LLC).
00: use cacheability control bits from GTT entry
01: data is not cached in LLC or MLC
10: data is cached in LLC but not MLC
11: data is cached in both LLC and MLC</td></tr>
<tr><td></td><td>23</td><td><b>ZLR bit</b>
Flag to indicate this is a zero length read, a read used to calculate a physical address for a write.</td></tr>
<tr><td></td><td>22:4</td><td><b>TAG</b>
Cycle identification TAG.</td></tr>
<tr><td></td><td>3:0</td><td><b>SRC ID</b>
Encoding of unit generating this cycle .
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0000b</td><td>CS_RD_SRCID</td></tr>
<tr><td>0001b</td><td>VF_RD_SRCID</td></tr>
<tr><td>0010b</td><td>ISC_SRCID</td></tr>
<tr><td>0011b</td><td>MT_SRCID</td></tr>
<tr><td>0100b</td><td>RCC_SRCID</td></tr>
<tr><td>0101b</td><td>HZARB_SRCID</td></tr>
<tr><td>0110b</td><td>RCZ_SRCID</td></tr>
</table></td></tr>
</table>

TLBPEND_SEC1 - Section 1 of TLBPEND Entry

		0111b	CTC_SRCID
		1000b	CS_WR_SRCID
		1001b	MBC_SRCID
		1010b	Reserved
		1011b	CS_RD_PWRCTX
		1100b	RC_R4WRCMP
		1101b	RESRVD2_SRCID
		1110b	RESRVD1_SRCID
		1111b	RESRVD0_SRCID

1.1.7.7 TLBPEND_SEC2 — Section 2 of TLBPEND entry

TLBPEND_SEC2 - Section 2 of TLBPEND Entry

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	04600h-046FCh

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

DWord	Bit	Description
0	31:14	**Reserved**
	13	**Big Page Attribute** This entry is using a 32K page.
	12:8	**Current Address** Format: GraphicsAddress[14:10] Bits 14:10 of the Virtual Address of the cycle.
	7:0	**PAT Entry** Location of Physical Address in Physical Address Table.

1.1.8 Configuration Registers for Graphic Arbiter

1.1.8.1 ZSHR — Depth/Early Depth TLB Partitioning Register

ZSHR - Depth/Early Depth TLB Partitioning Register

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000020
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	04050h

This register is used to determine the number of TLB entries from the total of 64 available to be used by the Depth partition of the TLB. The rest of the entries are used for the Early Depth/Stencil TLB.

DWord	Bit	Description	
0	31:6	**Reserved**	
		Format:	MBZ
	5:0	**Number of TLB Entries Out of 64 used for Depth TLB**	
		Default Value:	32
		The rest are be used for Early Depth/Stencil TLB. Default value is 32.	

1.1.8.2 Color/Depth Write FIFO Watermarks

CZWMRK - Color/Depth Write FIFO Watermarks

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	04060h

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

DWord	Bit	Description	
0	31:24	**Reserved**	
		Format:	MBZ
	23:18	**Color Wr Burst Size** This is the maximum size of the requests burst, from the last High Watermark trip, before reevaluating the High Watermark again.	
	17:16	**Reserved**	
		Format:	MBZ

CZWMRK - Color/Depth Write FIFO Watermarks

	15:12	**Color Wr FIFO High Watermark** This is the number of accumulated Color writes that will trigger a Burst of Z Writes.
	11:6	**Z Wr Burst Size** This is the maximum size of the requests burst, from the last High Watermark trip, before reevaluating the High Watermark again.
	5:4	**Reserved** Format: MBZ
	3:0	**Z Wr FIFO High Watermark** This is the number of accumulated Depth writes that will trigger a Burst of Z Writes.

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

PP_PFD[0:31] - PPGTT Page Fault Data Registers

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Address:	04580h

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

4580h-4583h: Fault Entry 0

...

45FCh-45FFh: Fault Entry 31

DWord	Bit	Description
0	31:12	**Fault Entry Page Address** Format: 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** Format: MBZ

1.1.9 Context Save Registers

1.1.9.1 SVG_CTX — SVG Context Save Register

SVG_CTX - SVG Context Save Register	
Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	WO
Size (in bits):	32
Address:	06FFCh
This register is used to send messages to enable context saving. This register may not be written from CPU.	

DWord	Bit	Description
0	31:16	**Masks** Format: Mask[15:0] A 1 in a bit in this field allows the modification of the corresponding bit in Bits 15:0
	15:1	**Reserved** Format: MBZ
	0	**Context Save Start** Default Value: 0h Format: Enable When a 1 is written to this bit with the mask bit set, it will kick off a context save. Once the save is complete the bit will be cleared.

1.1.9.2 SVL_CTX— SVL Context Save Register

SVL_CTX - SVL Context Save Register	
Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Address:	07FFCh
This register is used to send messages to enable context saving. This register may not be written from CPU.	

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

SVL_CTX - SVL Context Save Register

		Format:	MBZ
	0	**Context Save Start**	
		Default Value:	0h
		Format:	Enable
		When a 1 is written to this bit with the mask bit set, it will initiate a context save. Once the save is complete the bit will be cleared.	

1.1.9.3 WM_CTX— WM Context Save Register

WM_CTX - WM Context Save Register

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Address:	05FFCh

This register is used to send messages to enable context saving. This register may not be written from CPU.

DWord	Bit	Description	
0	31:16	**Masks**	
		Format:	Mask[15:0]
		A 1 in a bit in this field allows the modification of the corresponding bit in Bits 15:0	
	15:1	**Reserved**	
		Format:	MBZ
	0	**Context Save Start**	
		Default Value:	0h
		Format:	Enable
		When a 1 is written to this bit with the mask bit set, it will kick off a context save. Once the save is complete the bit will be cleared.	

1.1.9.4 SC_CTX— SC Context Save Register

SC_CTX - SC Context Save Register

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Address:	0E1FCh
Address:	0F1FCh
Name:	SC_CTX_SLICE1

This register is used to send messages to enable context saving. This register may not be written from CPU.

DWord	Bit	Description
0	31:16	**Masks** Format: Mask[15:0] A 1 in a bit in this field allows the modification of the corresponding bit in Bits 15:0
	15:5	**Reserved** Format: MBZ
	4	**Context Save Start(MMIO and NP State)** Format: Enable When a 1 is written to this bit with the mask bit set, it will kick off a context save. Once the save is complete the bit will be cleared.
	3:1	**Reserved** Format: MBZ
	0	**Context Save Start(MMIO Only)** Format: Enable When a 1 is written to this bit with the mask bit set, it will kick off a context save. Once the save is complete the bit will be cleared.

1.1.9.5 DM_CTX — DM Context Save Register

DM_CTX - DM Context Save Register

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32

DM_CTX - DM Context Save Register

Address:	0E0FCh
Address:	0F0FCh
Name:	DM_CTX_SLICE1

This register is used to send messages to enable context saving. This register may not be written from CPU.

DWord	Bit	Description
0	31:16	**Masks** Format: Mask[15:0] A 1 in a bit in this field allows the modification of the corresponding bit in Bits 15:0.
	15:1	**Reserved** Format: MBZ
	0	**Context Save Start** Default Value: 0h Format: Enable When a 1 is written to this bit with the mask bit set, it will initiate a context save. Once the save is complete the bit will be cleared.

1.1.9.6 SARB_CTX— SARB Context Save Register

SARB_CTX - SARB Context Save Register

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Address:	0B1FCh

This register is used to send messages to enable context saving. This register may not be written from CPU.

DWord	Bit	Description
0	31:16	**Masks** Format: Mask[15:0] A 1 in a bit in this field allows the modification of the corresponding bit in Bits 15:0
	15:1	**Reserved** Format: MBZ
	0	**Context Save Start** Default Value: 0h Format: Enable When a 1 is written to this bit with the mask bit set, it will kick off a context save. Once the save is complete the bit will be cleared.

1.1.9.7 VSC_CTX— VSC Context Save Register

VSC_CTX - VSC Context Save Register	
Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Address:	051FCh

This register is used to send messages to enable context saving. This register may not be written from CPU.

DWord	Bit	Description
0	31:16	**Masks** Format: Mask[15:0] A 1 in a bit in this field allows the modification of the corresponding bit in Bits 15:0
	15:1	**Reserved** Format: MBZ
	0	**Context Save Start** Default Value: 0h Format: Enable When a 1 is written to this bit with the mask bit set, it will kick off a context save. Once the save is complete, the bit will be cleared.

1.1.9.8 GPM_CTX— GPM Context Save Register

GPM_CTX - GPM Context Save Register	
Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Address:	080FCh

This register is used to send messages to enable context saving. This register may not be written from CPU.

DWord	Bit	Description
0	31:16	**Masks** Format: Mask[15:0] A 1 in a bit in this field allows the modification of the corresponding bit in Bits 15:0
	15:1	**Reserved** Format: MBZ
	0	**Context Save Start**

GPM_CTX - GPM Context Save Register

		Default Value:	0h
		Format:	Enable
		When a 1 is written to this bit with the mask bit set, it will initiate a context save. Once the save is complete the bit will be cleared.	

1.1.9.9 SOL_CTX— SOL Context Save Register

SOL_CTX - SOL Context Save Register

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Address:	052FCh

This register is used to send messages to enable context saving. This register may not be written from CPU.

DWord	Bit	Description
0	31:6	**Context Save Address/Offset** Default Value: 0h Format: Address This field specifies where the SOL context is to be saved. If **Power Context Save Mode** is disabled, then the value of this field is the virtual address of the location for SOL context to be saved. If **Power Context Save Mode** is enabled, then the value of this field is the offset into the power context image for SOL context to be saved.
	5:2	**Reserved** Format: MBZ
	1	**Power Context Save Mode** Default Value: 0h Format: Enable If set, then the save from SOL is for Power Context Save. If clear, then the save is for Ring Context Save.
	0	**Context Save Start** Default Value: 0h Format: Enable When a 1 is written to this bit with the mask bit set, it will kick off a context save. Once the save is complete, the bit will be cleared.

1.1.9.10 1.1.9.11 RING_BUFFER_HEAD_PREEMPT_REG

RING_BUFFER_HEAD_PREEMPT_REG - RING_BUFFER_HEAD_PREEMPT_REG

Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32

Address:	0214Ch
Name:	RCS RING_BUFFER_HEAD_PREEMPT_REG
ShortName:	RCS_RING_BUFFER_HEAD_PREEMPT_REG

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

This is a global register and context save restored as part of power context image.

Programming Notes

Programming Restriction:

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

DWord	Bit	Description
0	31:21	**Reserved** Format: MBZ
	20:2	**Preempted Head Offset** Format: U19 This field contains the Head pointer offset in the ring when the last MI_ARB_CHECK command was executed and caused the head pointer to move due to the UHPTR register being valid.
	1:0	**Ring/Batch Indicator** Format: Enabled

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

1.1.9.11 BB_ADDR_DIFF—Batch Buffer Address Difference Register

BB_ADDR_DIFF - Batch Address Difference Register	
Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Address:	02154h

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

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

DWord	Bit	Description
0	31:2	**Batch Buffer Address Difference** 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** Format: MBZ

1.1.10 Mode and Misc Ctrl Registers

1.1.10.1 GT4 Mode Control Register

B/D/F/Type:MBCunit

Address Offset:9038-903Bh

Default Value:0h

Access: RW; RO;

Size:32 bits

Bit	Access	Default Value	RST/PWR	Description
1:0	R/W	00b	Core	**GT4 Usage mode:** **00:** Non-GT4 **01:** GT4 is used in Alternate Frame rendering Mode (AFR) **10:** Basic Split Frame rendering Mode (SFR) **11:** Complex Split Frame rendering Mode (SFR w/ CBR)

Basic Split Frame Rendering is like CBR for all units except Windower. Windower should not be doing any checker boarding in basic SFR. The split programming should be done scissor range programming.

Complex Split Frame Rendering (aka CBR) is already defined in many DCNs

1.1.10.2 MI_MODE — Render Mode Register for Software Interface

<table>
<tr><th colspan="3">MI_MODE - Render Mode Register for Software Interface</th></tr>
<tr><td colspan="2">Register Space:</td><td>MMIO: 0/2/0</td></tr>
<tr><td colspan="2">Source:</td><td>RenderCS</td></tr>
<tr><td colspan="2">Default Value:</td><td>0x00000000</td></tr>
<tr><td colspan="2">Access:</td><td>R/W</td></tr>
<tr><td colspan="2">Size (in bits):</td><td>32</td></tr>
<tr><td colspan="2">Address:</td><td>0209Ch</td></tr>
<tr><td colspan="3">The MI_MODE register contains information that controls software interface aspects of the Memory Interface function.</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:16</td><td><b>Masks</b>
<table><tr><td>Format:</td><td>Mask[15:0]</td></tr></table>
A 1 in a bit in this field allows the modification of the corresponding bit in Bits 15:0</td></tr>
<tr><td></td><td>14</td><td><b>Async Flip Performance mode</b>
<table><tr><td></td><td></td></tr><tr><td>Format:</td><td>U1</td></tr></table>
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Performance mode enabled <b>[Default]</b></td><td>The stall of the flip event is in the windower</td></tr>
<tr><td>1h</td><td>Performance mode disabled</td><td>The stall of the flip event is in the command stream</td></tr></table>
<table><tr><th>Programming Notes</th></tr>
<tr><td>This bit should be set to '1' on all projects disabling Async Flip Performance mode.

When Async Flip Performance mode is enabled stall is in the Windower allowing the commands following the MI_WAIT_FOR_EVENT to be parsed by command streamer, this breaks the usage model of controlling the display message generation in display engine using MI_LOAD_REGISTER_IMMEDIATE commands from ring buffer.</td></tr></table></td></tr>
<tr><td></td><td>13</td><td><b>Flush Performance mode</b>
<table><tr><td></td><td></td></tr><tr><td>Format:</td><td>U1</td></tr></table>
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>run fast restore <b>[Default]</b></td><td>No NonPipelined SV flush.</td></tr>
<tr><td>1h</td><td>run slow legacy restore</td><td>With NonPipelined SV flush.</td></tr></table></td></tr>
<tr><td></td><td>11</td><td><b>Invalidate UHPTR enable</b>
<table><tr><td>Format:</td><td>Enable</td></tr></table></td></tr>
</table>

MI_MODE - Render Mode Register for Software Interface

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

10 **Reserved**

Format:	MBZ

9 **Rings Idle**

Format:	U1

Read Only Status bit

Value	Name	Description
0h	Not Idle **[Default]**	Parser not Idle or Ring Arbiter not Idle.
1h	Idle	Parser Idle and Ring Arbiter Idle.

Programming Notes
Writes to this bit are not allowed.

8 **Stop Rings**

Format:	U1

Value	Name	Description
0h	**[Default]**	Normal Operation.
1h		Parser is turned off and Ring arbitration is turned off.

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

6 **Vertex Shader Timer Dispatch Enable**

Format:	Enable

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

5 **Reserved**

Format:	MBZ

4 **Reserved**

Format:	MBZ

3:1 **Reserved**

Format:	MBZ

0 **Mask IIR disable**

Format:	Disable

Mask IIR disable. Nominally the Interrupt controller masks interrupts in the IIR register if an interrupt

MI_MODE - Render Mode Register for Software Interface

		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.

1.1.10.3 FF_Mode - Thread Mode Register

FF_Mode - Thread Mode Register

Register Space:	MMIO: 0/2/0
Source:	RenderCS 0x28A01010
Access:	R/W
Size (in bits):	32
Address:	020A0h

This register is used to program the FF shader Mode.

DWord	Bit	Description
0	31	**Reserved**

Format:	MBZ

DWord	Bit	Description
	30	**Reserved**

Format:	MBZ

DWord	Bit	Description
	29:26	**DS Hit Max Value**

Format:	U4

Description	Project
If the number of hits reaches the DS Hit Max Value and there is a pending miss to be dispatched, the DS will dispatch the pending miss vertex as a single dispatch.	
Programming the value beyond the range will have undefined behavior.	

Value	Name	Project
10	**[Default]**	
[1,11]		

DWord	Bit	Description
	25:20	**VS Hit Max Value**

Format:	U6

Description	Project
If the number of hits reaches the VS Hit Max Value and there is a pending miss to be dispatched, the VS will dispatch the pending miss vertex as a single dispatch.	
Programming the value beyond the range will have undefined behavior.	

FF_Mode - Thread Mode Register

Value	Name	Project
10	**[Default]**	
[1,58]		

19 **DS Reference Count Full Force Miss Enable**

Project:	
Format:	Enable

Value	Name	Description
0b	**[Default]**	On a hit to the DS cache and the associated handle’s reference count is full then stall until a derefernce.
1b		On a hit to the DS cache and the associated handle’s reference count is full then force the cycle as a miss and allocate a new handle.

18:17 **TS Thread Dispatch Mode**

Format:	U2

Value	Name	Description
0h	Load Balanced **[Default]**	Thread Dispatch will load balance the half slices of the threads. Note: this will cause possible corruption if input handles are reused due to instancing or topologies that reuse vertices(i.e. strips and fans)
1h	Half Slice 0	All threads will be dispatched to Half Slice 0.
2h	Half Slice 1	All threads will be dispatched to Half Slice 1.
3h	Reserved	

16 **TS Thread Dispatch Override Enable**

Format:	Enable

Value	Name	Description
0h	Disable **[Default]**	Hardware will decide which half slice the thread will dispatch.
1h	Enable	The value in the TS Thread Dispatch Mode will be used for dispatch.

15 **VS Reference Count Full Force Miss Enable**

Format:	U1

Value	Name	Description
[0,1]		
0b	**[Default]**	On a hit to the VS cache and the associated handle's reference count is full then stall until a derefernce.
1b		On a hit to the VS cache and the associated handle's reference count is full then force the cycle as a miss and allocate a new handle.

14:13 **VS Thread Dispatch Mode**

Format:	U2

FF_Mode - Thread Mode Register

Value	Name	Description
0h	Load Balanced **[Default]**	Thread Dispatch will load balance the half slices of the threads. Note: this will cause possible corruption if input handles are reused due to instancing or topologies that reuse vertices (i.e. strips and fans)
1h	Half Slice 0	All threads will be dispatched to Half Slice 0.
2h	Half Slice 1	All threads will be dispatched to Half Slice 1.
3h	Reserved	

12 **VS Thread Dispatch Override Enable**

Format:	Enable

Value	Name	Description
0h	Disable	Hardware will decide which half slice the thread will dispatch.
1h	Enable **[Default]**	The value in the VS Thread Dispatch Mode will be used for dispatch.

11:7 **Reserved**

Format:	MBZ

6:5 **DS Thread Dispatch Mode**

Format:	U2

Value	Name	Description
0h	Load Balanced **[Default]**	Thread Dispatch will load balance the half slices of the threads. Note: this will cause possible corruption if input handles are reused due to instancing or topologies that reuse vertices (i.e., strips and fans).
1h	Half Slice 0	All threads will be dispatched to Half Slice 0.
2h	Half Slice 1	All threads will be dispatched to Half Slice 1.
3h	Reserved	

4 **DS Thread Dispatch Override Enable**

Format:	Enable

Value	Name	Description
0h	Disable	Hardware will decide which half slice the thread will dispatch.
1h	Enable **[Default]**	The value in the DS Thread Dispatch Mode will be used for dispatch.

3:0 **Reserved**

Format:	MBZ

1.1.10.4 GFX_MODE – Graphics Mode Register

GFX_MODE - Graphics Mode Register

Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000800
Size (in bits):	32
Trusted Type:	1
Address:	0229Ch

Description	Project
This register contains a control bit for the new 2-level PPGTT functions.	
DefaultValue = 00002800h	

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:16</td><td><b>Mask Bits</b>
<table><tr><td>Format:</td><td>Mask[15:0]</td></tr></table>
Must be set to modify corresponding bit in Bits 15:0. (All implemented bits)</td></tr>
<tr><td></td><td>14</td><td><b>Reserved</b>
<table><tr><td>Project:</td><td></td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>13</td><td><b>Flush TLB invalidation Mode</b>
<table><tr><td></td><td></td></tr><tr><td>Format:</td><td>U1</td></tr></table>
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, to pipe_control commands which have the TLB invalidation bit set and sync flushes. If disabled, the TLB caches are flushed for every full flush of the pipeline.</td></tr>
<tr><td></td><td>12</td><td><b>Reserved</b>
<table><tr><td>Project:</td><td>All</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11</td><td><b>Replay Mode</b>
<table><tr><td></td><td></td></tr><tr><td>Format:</td><td>U1 Context Switch Granularity</td></tr></table>
This field controls the granularity of the replay mechanism when coming back into a previously preempted context.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th><th>Project</th></tr>
<tr><td>0h</td><td>mid-triangle preemption</td><td>Super span Level. Pipeline is not flushed. This implies commands parsed are executed speculatively and may not complete before a context switch.</td><td></td></tr>
<tr><td>1h</td><td>mid-cmdbuffer preemption <b>[Default]</b></td><td>Drawcall Level. Pipeline is flushed before switching to the next context. Commands parsed are commited to completing before a context switch</td><td></td></tr>
</table></td></tr>
</table>

GFX_MODE - Graphics Mode Register

	Programming Notes		
	A fixed function pipe flush is required before modifying this fieldUnless pre-emption at a mid-triangle is required the bit must be set.		
10	**Reserved**		
	Project:	All	
	Format:	MBZ	
9	**Per-Process GTT Enable**		
	Format:	Enabled	
	Per-Process GTT Enable		
	Value	**Name**	**Description**
	0h	PPGTT Disable **[Default]**	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.
	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)
8	**Reserved**		
	Format:	MBZ	
7	**Reserved**		
	Format:	MBZ	
6:1	**Reserved**		
	Format:	MBZ	
0	**Reserved**		
	Format:	MBZ	

1.1.10.5 GT_MODE – GT Mode Register

GT_MODE - GT Mode Register

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000200
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	07008h

This Register is used to control the 6EU and 12EU configuration for GT.

Write 0x01FF01FF to this register enables the 6EU mode.

RegisterType = MMIO_SVL

DWord	Bit	Description
0	31:16	**Mask Bits** Format: Mask[15:0] Must be set to modify corresponding bit in Bits 15:0. (All implemented bits)
	15	**Reserved** Format: MBZ
	14:11	**Reserved** Format: MBZ
	10	**Reserved** Format: MBZ
	9	**WIZ Hashing Mode High Bit** 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
0h		8x8 Checkerboard hashing
1h	**[Default]**	8x4 Checkerboard hashing
2h		16x4 Checkerboard hashing
3h		Reserved

DWord	Bit	Description
	7	**WIZ Hashing Mode** Project: Format: U1

GT_MODE - GT Mode Register

DWord	Bit	Description

Description	Project
This field configures the Hashing mode in Windower. This field is don’t care if the Hashing Disable bit is set.	
The WIZ Hashing Mode High Bit field is combined with this field to enable additional modes.	

DWord	Bit	Description
	2	**Reserved**

Format:	MBZ

DWord	Bit	Description
	0	**Reserved**

1.1.10.6 Cache_Mode_0— Cache Mode Register 0

Cache_Mode_0 - Cache Mode Register 0

Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
	0x00000004
Access:	R/W
Size (in bits):	32
Address:	07000h

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.

RegisterType = MMIO_SVL

DWord	Bit	Description
0	31:16	**Masks**

Format:	Mask[15:0]

A 1 in a bit in this field allows the modification of the corresponding bit in Bits 15:0.

DWord	Bit	Description
	15	**Sampler L2 Disable**

Format:	Disable

Value	Name	Description
0h	**[Default]**	Sampler L2 Cache Enabled.
1h		Sampler L2 Cache Disabled all accesses are treated as misses.

DWord	Bit	Description
	14:12	**Reserved**

Format:	MBZ

Cache_Mode_0 - Cache Mode Register 0

11 **Reserved**

Format:	MBZ

10 **Reserved**

Format:	MBZ

9 **Sampler L2 TLB Prefetch Enable**

Value	Name	Description
0h	**[Default]**	TLB Prefetch Disabled
1h		TLB Prefetch Enabled

7:6 **Sampler L2 Request Arbitration**

Format:	U2

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

5 **STC Eviction Policy**

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.

Note: If this bit is set to "1", bit 7 of 0x7010h must also be set to "1"

4 **RCC Eviction Policy**

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.

Note: If this bit is set to "1", bit 7 of 0x7010h must also be set to "1"

3 **Hierarchical Z Disable**

Mask:	MMIO(0x2120)#19
Format:	U1

Value	Name
0h	Enable
1h	Disable

2 **Hierarchical Z RAW Stall Optimization Disable**

Format:	U1

The Hierarchical Z RAW Stall Optimization allows non-overlapping polygons in the same 8x4 pixel/sample area to be processed without stalling waiting for the earlier ones to write to Hierarchical Z

Cache_Mode_0 - Cache Mode Register 0

buffer.

Value	Name	Description
0h	Enable	Enables the hierarchical Z RAW Stall Optimization.
1h	Disable **[Default]**	Disables the hierarchical Z RAW Stall Optimization.

DWord	Bit	Description
	1	**Disable clock gating in the pixel backend** Format: Disable
	0	**Render Cache Operational Flush Enable** Format: Enable

Format:	Disable

Format:	Enable

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

Errata	Description	Project
Erratum	This bit must be set to ‘0’ (Disable)	

1.1.10.7 Cache_Mode_1— Cache Mode Register 1

Cache_Mode_1 - Cache Mode Register 1

Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000180
Access:	Read/32 bit Write Only
Size (in bits):	32

Address:	07004h

Description	Project
RegisterType: MMIO_SVL	
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.	

DWord	Bit	Description
0	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
	14	**Reserved**

Cache_Mode_1 - Cache Mode Register 1

Format:	MBZ

12 **HIZ Eviction Policy**

Project:	All
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	**[Default]**	Non-LRA eviction Policy	All
1h		LRA eviction Policy	All

Programming Notes	Project
If this bit is set to "1", bit 3 of 0x7010h must also be set to "1"	

11 **Reserved**

Format:	MBZ

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

Project:	All
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
00b	None	No XOR.	All
01b	Scheme 1	New_set_mask[3:0] = Tiled_address[16:13]. New_set[3:0] less than or = 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

10b	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].
			All

Cache_Mode_1 - Cache Mode Register 1

		New_set[3:0] less than or = 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.	

11b	Scheme 3 **[Default]**	New_set_mask[3] = Tiled_address[22] ^ 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] less than or = 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 **Pixel Backend sub-span collection Optimization Disable**

Format:	Disable

Value	Name	Description
0h	**[Default]**	Enables two contiguous quads to be collected as 4X2 access for RCZ interface. This allows for less bank collision and less RAM power on RCZ.
1h		Disables this optimization and therefore only one valid sub-span is sent to RCZ on the 4X2 interface.

Programming Notes	Project
This bit must be set.	

5 **MCS Cache Disable**

Format:	Disable

For Programming restrictions please refer to the 3D Pipeline.

Value	Name	Description

Cache_Mode_1 - Cache Mode Register 1

0h	**[Default]**	MCS cache enabled. It allows RTs with MCS buffer enabled to be rendered using either MSAA compression for MSRT OR with color clear feature for non MSRT.
1h		MCS cache is disabled. Hence no MSAA compression for MSRT and no color clear for non-MSRT.

4 **Reserved**

Format:	MBZ

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

Format:	Disable

Description	Project
This bit must always be reset to "0".	

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

2:1 **Reserved**

1.1.10.8 GAFS_MODE — Mode Register for GAFS

GAFS_MODE - Mode Register for GAFS

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	0212Ch

DWord	Bit	Description
0	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.
	15:10	**Reserved** Format: MBZ
	8:2	**Reserved** Format: MBZ
	0	**Selection of Arbitration for GAFS**

GAFS_MODE - Mode Register for GAFS

		Format:	MBZ
		GAFS data return policy from FFROB is a round-robin. This bit freezes the round robin to FF pipeline order.	

1.1.10.9 INSTPM—Instruction Parser Mode Register

INSTPM - Instruction Parser Mode Register

Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	020C0h

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.

DWord	Bit	Description
0	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.
	14:13	**Reserved** Format: MBZ
	11	**CLFLUSH Toggle** Format: U1 This bit changes polarity each time the MI_CLFLUSH command completes.

INSTPM - Instruction Parser Mode Register

10 **Reserved**

Format:	MBZ

9 **TLB Invalidate**

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 up to GFX driver to explicitly invalidate TLBs post reset.

8 **Memory Sync Enable**

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**

Format:	Enable

Description	Project
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 commands. See section 3.2.2 for a list of these commands.	
[This bit should be programmed to 1.	

Value	Name
0b	**[Default]**
1b	

6 **CONSTANT_BUFFER Address Offset Disable**

Format:	Disable

When this bit is clear, the 3DSTATE_CONSTANT_* Buffers' Starting Address is used as a DynamicStateOffset. That is, it serves as an offset from the Dynamic State Base Address. 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.

5 **Sync Flush Enable**

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 Notes	Project
- 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	

INSTPM - Instruction Parser Mode Register

	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.	
	Workaround : Write 0x2050[31:0] = 0x00010001 (disable sequence) Write 0x2700[31:0] = 0x00000000 (Wake up CS but don't do anything) Poll 0x22AC[3:0] = 0 (Guarantees render pipe is awake) VT-d request(Sync Flush) (Normal VT-d cycles(Replace with Sync Flush Steps) Write 0x2050[31:0] = 0x00010000 (Enable sequence (to enter RC6))	
3	**Media Instruction Disable**	
	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**	
	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**	
	Format:	Disable
0	**Texture Palette Load Instruction Disable**	
	Format:	U1
	This bit instructs the Renderer instruction parser to parse and error-check Texture Palette Load instructions, but not execute them. Format = Disable	

1.1.10.10 EXCC—Execute Condition Code Register

EXCC - Execute Condition Code Register

Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W,RO
Size (in bits):	32
Trusted Type:	1
Address:	02028h

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

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

DWord	Bit	Description
0	31:16	**Mask Bits** Format: Mask[15:0] These bits 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** Format: MBZ
	11	**Pending Indirect State Dirty Bit** 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** 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** 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).

1.1.10.11 NOPID — NOP Identification Register

NOPID - NOP Identification Register	
Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000
Size (in bits):	32
Trusted Type:	1
Address:	02094h

Description	Project
Access: RW	
The NOPID register contains the Noop Identification value specified by the last MI_NOOP instruction that enabled this register to be updated.	

DWord	Bit	Description
0	31:22	**Reserved** Format: MBZ

1.1.10.12 FBC RT BASE ADDRESS REGISTER

FBC_RT_BASE_ADDR_REGISTER - FBC_RT_BASE_ADDR_REGISTER	
Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	Read/32 bit Write Only
Size (in bits):	32
Address:	07020h

This Register is saved and restored as part of Context.

DWord	Bit	Description
0	31:12	**FBC RT Base Address** Format: PPGraphicsAddress[31:12] 4KB aligned Base Address as mapped in the PPGTT or in the GGTT for the render target. Must be set to modify corresponding data bit. Reads to this field returns zero. 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.
	11:2	**Reserved** Format: MBZ
	1	**FBC Front Buffer Target** Format: Enable

FBC_RT_BASE_ADDR_REGISTER - FBC_RT_BASE_ADDR_REGISTER

Value	Name	Description	Project
0h	**[Default]**	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.	
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.	

	0	**PPGTT Render Target Base Address Valid for FBC**

Access:	None
Exists If:	Always
Format:	Enable
Format:	GraphicsAddress[31:0]U32

Value	Name	Description
0h	**[Default]**	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.

Programming Notes
Workaround : Do not enable Render Command Streamer tracking for FBC. Instead insert a LRI to address 0x50380 with data 0x00000004 after the PIPE_CONTROL that follows each render submission.

1.1.10.13 RVSYNC – Render/Video Semaphore Sync Register

RVSYNC - Render/Video Semaphore Sync Register

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	02040h

This register is written by VCS, read by CS.

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

1.1.10.14 RBSYNC – Render/Blitter Semaphore Sync Register

RBSYNC - Render/Blitter Semaphore Sync Register	
Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	02044h

This register is written by BCS, read by CS.

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

1.1.11 RINGBUF — Ring Buffer Registers

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

1.1.11.1 RING_BUFFER_TAIL

RING_BUFFER_TAIL - Ring Buffer Tail	
Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Access:	R/W
Address:	02030h
Name:	RCS Ring Buffer Tail
ShortName:	RCS_RING_BUFFER_TAIL
Address:	12030h
Name:	VCS Ring Buffer Tail
ShortName:	VCS_RING_BUFFER_TAIL
Address:	22030h
Name:	BCS Ring Buffer Tail
ShortName:	BCS_RING_BUFFER_TAIL

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

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

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

RING_BUFFER_TAIL - Ring Buffer Tail

empty.

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

1.1.11.2 RING_BUFFER_HEAD

RING_BUFFER_HEAD - Ring Buffer Head

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

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

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

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

DWord	Bit	Description	
0	31:21	**Wrap Count**	
		Format:	U11 count of ring buffer wraps

RING_BUFFER_HEAD - Ring Buffer Head		
		This field is incremented by 1 whenever the **Head Offset** wraps from the end of the buffer back to the start (i.e., whenever it wraps back to 0). Appending this field to the **Head Offset** field effectively creates a virtual 4GB Head "Pointer" which can be used as a tag associated with instructions placed in a ring buffer. The Wrap Count itself will wrap to 0 upon overflow.
	20:2	**Head Offset** Format: GraphicsAddress[20:2] DWord Offset This field indicates the offset of the *next* instruction DWord to be parsed. Software will initialize this field to select the first DWord to be parsed once the RB is enabled. (Writing the Head Offset while the RB is enabled is UNDEFINED). Subsequently, the device will increment this offset as it executes instructions – until it reaches the QWord specified by the **Tail Offset**. At this point the ring buffer is considered "empty". **Programming Notes** A RB can be enabled empty or containing some number of valid instructions.
	1	**Reserved** Format: MBZ
	0	**Wait for Condition Indicator** Source: RenderCS This is a read only value used to indicate whether or not the command streamer is currently waiting for a conditional code to be cleared from 0x2028
	0	**Reserved** Source: BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS Format: MBZ

1.1.11.3 RING_BUFFER_START

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

RING_BUFFER_START - Ring Buffer Start

ShortName:	BCS_RING_BUFFER_START

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

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

1.1.11.4 RING_BUFFER_CONTROL

RING_BUFFER_CTL - Ring Buffer Control

Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Access:	R/W
Address:	0203Ch
Name:	RCS Ring Buffer Control
ShortName:	RCS_RING_BUFFER_CTL
Address:	1203Ch
Name:	VCS Ring Buffer Control
ShortName:	VCS_RING_BUFFER_CTL
Address:	2203Ch
Name:	BCS Ring Buffer Control
ShortName:	BCS_RING_BUFFER_CTL

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

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

DWord	Bit	Description
0	31:21	**Reserved**

RING_BUFFER_CTL - Ring Buffer Control

Format:	MBZ

20:12 **Buffer Length**

Format:	U9-1 in 4 KB pages – 1

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]

Value	Name	Description
0		1 page = 4 KB
1FFh		512 pages = 2 MB

11 **RBWait**

Indicates that this ring has executed a WAIT_FOR_EVENT instruction and is currently waiting. Software can write a “1” to clear this bit, write of “0” has no effect. When the RB is waiting for an event and this bit is cleared, the wait will be terminated and the RB will be returned to arbitration.

10 **Semaphore Wait**

Description	Project
Indicates that this ring has executed a MI_SEMAPHORE_MBOX instruction with register compare and is currently waiting.	

9 **Reserved**

Format:	MBZ

8 **Reserved**

Source:	RenderCS, BlitterCS
Format:	MBZ

8 **Disable Register Accesses**

Source:	VideoCS, VideoCS2, VideoEnhancementCS

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

7:3 **Reserved**

Format:	MBZ

2:1 **Automatic Report Head Pointer**

Source:	BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS

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

Value	Name	Description
0	MI_AUTOREPORT_OFF	Automatic reporting disabled

RING_BUFFER_CTL - Ring Buffer Control

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

2:1 **Reserved**

Source:	RenderCS
Format:	MBZ

0 **Ring Buffer Enable**

Format:	Enable

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

Programming Notes	Project
SW should follow the below programming notes while enabling render engine's ring buffer for the first time, this would be coming out of boot, standby, hibernate or reset. SW should set the Force Wakeup bit to prevent GT from entering C6. SW should dispatch workload (dummy) to initialize render engine with default state such that any context switches that occur subsequently (Power Save) will save and restore coherent device state. Indirect pointers used in 3D states should point to valid graphics surface existing n memory. PP_DCLV followed by PP_DIR_BASE register should be programmed as part of nitialization workload if PPGTT is enabled in GFX_MODE register. Once the render engine is programmed with valid state and the configuration, Force Wakeup bit should be reset to enable C6 entry.	

1.1.11.5 UHPTR — Pending Head Pointer Register

UHPTR - Pending Head Pointer Register

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

Address:	02134h
Name:	RCS Pending Head Pointer Register
ShortName:	RCS_UHPTR
Address:	12134h
Name:	VCS Pending Head Pointer Register
ShortName:	VCS_UHPTR
Address:	22134h
Name:	BCS Pending Head Pointer Register

UHPTR - Pending Head Pointer Register

ShortName:	BCS_UHPTR

Programming Notes

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

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

Value	Name	Description
0	InValid	No valid updated head pointer register, resume execution at the current location in the ring buffer
1	Valid	Indicates that there is an updated head pointer programmed in this register

1.1.12 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.

1.1.12.1 PR_CTR_CTL—Render Watchdog Counter Control

PR_CTR_CTL - Render Watchdog Counter Control	
Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Address:	02178h

DWord	Bit	Description
0	31	**Count Select** 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** 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.

1.1.12.2 PR_CTR_THRSH—Render Watchdog Counter Threshold

PR_CTR_THRSH - Render Watchdog Counter Threshold	
Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00150000
Access:	R/W
Size (in bits):	32
Address:	0217Ch

DWord	Bit	Description
0	31:0	**Counter logic Threshold** Default Value: 00150000h 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.

1.1.12.3 PR_CTR—Render Watchdog Counter

PR_CTR - Render Watchdog Counter	
Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Address:	02190h

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

1.1.13 Interrupt Control Registers

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

Bit Definition for Interrupt Control Registers	
Source:	RenderCS
Default Value:	0x00000000

DWord	Bit	Description
0	31:12	**Reserved** Format: MBZ Reserved for other command streamers - can not be allocated by main command streamer.
	11:10	**Reserved** Format: MBZ These bits may be assigned to interrupts on future products/steppings.
	9	**Performance Monitoring Buffer Half-Full Interrupt** For internal trigger (timerbased) based reporting, if the report buffer crosses half full limit, this interrupt is generated.
	8	**Context Switch Interrupt**
	7	**Page Fault** Project: All

Bit Definition for Interrupt Control Registers

<table>
<tr><td></td><td></td><td><table><tr><th>Description</th><th>Project</th></tr><tr><td>This bit is set whenever there is a pending GGTT/PPGTT (page or directory) fault in Render command streamer.</td><td></td></tr></table></td></tr>
<tr><td></td><td>6</td><td><b>Timeout Counter Expired</b>
Set when the render pipe timeout counter (0x02190) has reached the timeout thresh-hold value (0x0217c).</td></tr>
<tr><td></td><td>5</td><td><b>Reserved</b><table><tr><td></td><td></td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>4</td><td><b>PIPE_CONTROL Notify Interrupt</b>
The Pipe Control packet (Fences) specified in 3D pipeline document may optionally generate an Interrupt. The Store QW associated with a fence is completed ahead of the interrupt.</td></tr>
<tr><td></td><td>3</td><td><b>Render Command Parser Master Error</b>

When this status bit is set, it indicates that the hardware has detected an error. It is set by the device upon an error condition and cleared by a CPU write of a one to the appropriate bit contained in the Error ID register followed by a write of a one to this bit in the IIR. Further 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.

<b>Page Table Error:</b> Indicates a page table error.

<b>Instruction Parser Error:</b> The Render Instruction Parser encounters an error while parsing an instruction.</td></tr>
<tr><td></td><td>2</td><td><b>Sync Status</b><table><tr><td></td><td></td></tr></table>
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). <b>It is the driver's responsibility to clear this bit before the next sync flush with HWSP write enabled.</b></td></tr>
<tr><td></td><td>0</td><td><b>Render Command Parser User Interrupt</b>
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.</td></tr>
</table>

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	**Media Decode Pipeline Counter Exceeded Notify Interrupt:** The counter threshold for the execution of the media pipeline is exceeded. Driver needs to attempt hang recovery.	Not supported to be unmasked
5	**L3 Parity interrupt**	
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

1.1.13.1 HWSTAM — Hardware Status Mask Register

HWSTAM - Hardware Status Mask Register

Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0xFFFFFFFF
Access:	R/W,RO
Size (in bits):	32
Trusted Type:	1
Address:	02098h

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

Programming Notes

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

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

1.1.13.2 IMR—Interrupt Mask Register

IMR - Interrupt Mask Register	
Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0xFFFFFFFF
Access:	R/W,RO
Size (in bits):	32
Address:	020A8h

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

DWord	Bit	Description
0	31:0	**Interrupt Mask Bits**
		Format: InterruptMask[32] Refer to the Interrupt Control Register section for bit definitions.
		This field contains a bit mask which selects which interrupt bits (from the ISR) are reported in the IIR. Reserved bits in the Interrupt Control Register are RO.

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

1.1.13.3 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:

Hardware-Detected Error Bit Definitions

Source:	RenderCS
Default Value:	0x00000000

<table>
<tr><th>DWord</th><th>Bit</th><th colspan="3">Description</th></tr>
<tr><td>0</td><td>31:3</td><td colspan="3"><b>Reserved</b></td></tr>
<tr><td></td><td></td><td colspan="2">Format:</td><td>MBZ</td></tr>
<tr><td></td><td>2</td><td colspan="3"><b>Reserved</b></td></tr>
<tr><td></td><td></td><td colspan="2"></td><td></td></tr>
<tr><td></td><td></td><td colspan="2">Format:</td><td>MBZ</td></tr>
<tr><td></td><td>1</td><td colspan="3"><b>Reserved</b></td></tr>
<tr><td></td><td></td><td colspan="2">Format:</td><td>MBZ</td></tr>
<tr><td></td><td>0</td><td colspan="3"><b>Instruction Error</b>
This bit is set when the Renderer Instruction Parser detects an error while parsing an instruction.
Instruction errors include:

Client ID value (Bits 31:29 of the Header) is not supported (only MI, 2D and 3D are supported).
Defeatured MI Instruction Opcodes:</td></tr>
<tr><td></td><td></td><td><b>Value</b></td><td><b>Name</b></td><td><b>Description</b></td></tr>
<tr><td></td><td></td><td>1</td><td></td><td>Instruction Error detected</td></tr>
<tr><td></td><td></td><td colspan="3"><b>Programming Notes</b></td></tr>
<tr><td></td><td></td><td colspan="3">This error indications cannot be cleared except by reset (i.e., it is a fatal error).</td></tr>
</table>

1.1.13.3.1 EIR — Error Identity Register

EIR - Error Identity Register

Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W,RO
Size (in bits):	32
Address:	020B0h

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

<table>
<tr><th>DWord</th><th>Bit</th><th colspan="2">Description</th></tr>
<tr><td>0</td><td>31:16</td><td colspan="2"><b>Reserved</b></td></tr>
<tr><td></td><td></td><td>Format:</td><td>MBZ</td></tr>
<tr><td></td><td>15:0</td><td colspan="2"><b>Error Identity Bits</b></td></tr>
<tr><td></td><td></td><td>Format:</td><td>Array of Error condition bits See the table titled Hardware-Detected Error Bits.</td></tr>
<tr><td></td><td></td><td colspan="2">This register contains the persistent values of ESR error status bits that are unmasked via the EMR register. (See Table Table 3-3. Hardware-Detected Error Bits). The logical OR of all (defined) bits in</td></tr>
</table>

EIR - Error Identity Register

this register is reported in the Master Error bit of the Interrupt Status Register. In order to clear an error condition, software must first clear the error by writing a 1 to the appropriate bit(s) in this field. If required, software should then proceed to clear the Master Error bit of the IIR. Reserved bits are RO.

Value	Name
1h	Error occurred

Programming Notes
Writing a 1 to a set bit will cause that error condition to be cleared. However, neither the Page Table Error bit (Bit 4) nor the Instruction Error bit (Bit 0) can be cleared except by reset (i.e., it is a fatal error).

1.1.13.3.2 EMR—Error Mask Register

EMR - Error Mask Register

Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x000000FF
Access:	R/W,RO
Size (in bits):	32
Address:	020B4h

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.

DWord	Bit	Description
0	31:8	**Reserved**

Format:	Must Be One

Programming Notes
These bits are not implemented in HW and must be set to ’1’

DWord	Bit	Description
	7:0	**Error Mask Bits**

Format:	Array of error condition mask bits See the table titled Hardware-Detected Error Bits.

This register contains a bit mask that selects which error condition bits (from the ESR) are reported in the EIR.

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

1.1.13.3.3 ESR—Error Status Register

ESR - Error Status Register	
Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32
Address:	020B8h

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.

DWord	Bit	Description
0	31:16	**Reserved** Format: MBZ
	15:0	**Error Status Bits** Format: Array of error condition bits See the table titled Hardware-Detected Error Bits. This register contains the non-persistent values of all hardware-detected error condition bits. **Value** — **Name** 1h — Error Condition Detected

1.1.14 Logical Context Support

1.1.14.1 BB_ADDR — Batch Buffer Head Pointer

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

BB_ADDR - Batch Buffer Head Pointer Register

Name:	VECS Batch Buffer Head Pointer Register
ShortName:	VECS_BB_ADDR

Address:	22140h
Name:	BCS Batch Buffer Head Pointer Register
ShortName:	BCS_BB_ADDR

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

Programming Notes

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

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:3</td><td><b>Batch Buffer Head Pointer</b>
<table>
<tr><td></td><td></td></tr>
<tr><td>Source:</td><td>BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS</td></tr>
<tr><td>Format:</td><td>GraphicsAddress[31:3]</td></tr>
</table>
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.</td></tr>
<tr><td></td><td>31:2</td><td><b>Batch Buffer Head Pointer</b>
<table>
<tr><td></td><td></td></tr>
<tr><td>Source:</td><td>RenderCS</td></tr>
<tr><td>Format:</td><td>GraphicsAddress[31:2]</td></tr>
</table>
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.</td></tr>
<tr><td></td><td>2</td><td><b>Reserved</b>
<table>
<tr><td></td><td></td></tr>
<tr><td>Source:</td><td>BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS</td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>1</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>0</td><td><b>Valid</b>
<table>
<tr><td>Format:</td><td>U1</td></tr>
</table>
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Invalid <b>[Default]</b></td><td>Batch buffer Invalid</td></tr>
<tr><td>1h</td><td>Valid</td><td>Batch buffer Valid</td></tr>
</table></td></tr>
</table>

1.1.14.2 BB_STATE – Batch Buffer State Register

BB_STATE - Batch Buffer State Register

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	RO
Size (in bits):	32

Address:	02110h
Name:	RCS Batch Buffer State Register
ShortName:	RCS_BB_STATE

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

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

This register is saved and restored with context.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:9</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>8</td><td><b>Reserved</b><table><tr><td></td><td></td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7</td><td><b>Reserved</b><table><tr><td></td><td></td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>5</td><td><b>Address Space Indicator</b><table><tr><td></td><td></td></tr></table><table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>GGTT <b>[Default]</b></td><td>This batch buffer is located in GGTT memory</td></tr><tr><td>1h</td><td>PPGTT</td><td>This batch buffer is located in PPGTT memory.</td></tr></table></td></tr>
<tr><td></td><td>3:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
</table>

1.1.14.3 CCID—Current Context Register

CCID - Current Context Register

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Address:	02180h

This register contains the current logical rendering context address associated with the ring buffer in ring buffer mode of scheduling. This register contents are not valid in Exec-List mode of scheduling.

Programming Notes

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.

DWord	Bit	Description
0	31:12	**Logical Render Context Address (LRCA)** Default Value: 0h Format: GraphicsAddress[31:11] This field contains the 4 KB-aligned Graphics Memory Address of the current Logical Rendering Context. Bit 11 MBZ.
	11:10	**Reserved** Format: MBZ
	8	**Reserved** Format: Must Be One
	7:4	**Reserved** Format: MBZ
	3	**Extended State Save Enable** 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** 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.
	1	**Reserved** Format: MBZ
	0	**Valid** Format: U1

Value	Name	Description
0h	Invalid **[Default]**	The other fields of this register are invalid. A switch away from the context will not invoke a context save operation.

CCID - Current Context Register

		1h	Valid	The other fields of this register are valid, and a switch from the context will invoke the normal context save/restore operations.

1.1.14.4 CXT_SIZE—Context Sizes

CXT_SIZE - Context Sizes

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x48A7B8CD
Access:	R/W
Size (in bits):	32
Trusted Type:	1
Address:	021A8h

The actual size of a logical rendering context is the amount of data stored/restored during a context switch and is measured in 64B cache lines. This register will be power context save/restored. Note that this register will default to the correct value, so software should not have to modify it.

DWord	Bit	Description
0	31:25	**Power Context Size** Default Value: 24h This field indicates the Power context data that need to be save restored.
	24:22	**Ring Context Size** Default Value: 2h This field indicates the Ring context data that need to be save restored.
	21:16	**Render Context Size** Default Value: 27h This field indicates the render context data that need to be save restored when extended mode is not enabled for a context.
	15:9	**Extended Context Size** Default Value: 5Ch This field indicates the render context data that need to be save restored when extended mode is enabled for a context. Note: Render context is subset of this context.
	8:6	**GT1 Mode** Default Value: 3h This field indicates the amount of data that need not be save/restored from render context in GT1 mode. Note: This is the amount of data not save/restored from TDL and SC in GT1 mode.
	5:0	**VF State Context Size** Default Value: Dh This field indicates the amount of VF unit data context save/restored in cachelines.

1.1.14.5 MTCH_CID_RST – Matched Context ID Reset Register

MTCH_CID_RST - Matched Context ID Reset Register

Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Address:	0222Ch

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.

DWord	Bit	Description
0	31:12	**Match Context ID** Format: U20 Contains the context ID to be compared with the currently running context ID.
	11:2	**Reserved** Format: MBZ
	1	**Match** 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** 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)

1.1.14.6 SYNC_FLIP_STATUS – Wait for Event and Display Flip Flags Register

SYNC_FLIP_STATUS - Wait For Event and Display Flip Flags Register

Register Space:	MMIO: 0/2/0
Project:	All
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32

Address:	022D0h
Name:	RCS Wait For Event and Display Flip Flags Register
ShortName:	RCS_SYNC_FLIP_STATUS

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.

Programming Notes

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

DWord	Bit	Description
0	31	**Reserved** Format: MBZ
	30	**Display Plane A Asyncronous Display Flip Pending** 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** 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** 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** Format: MBZ
	26	**Display Plane B Asyncronous Display Flip Pending** 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

SYNC_FLIP_STATUS - Wait For Event and Display Flip Flags Register

		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** 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** 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** Source: | RenderCS 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** 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** 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.
	20	**Display Sprite A Syncronous Flip Pending Wait Enable** 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** Format: | MBZ

SYNC_FLIP_STATUS - Wait For Event and Display Flip Flags Register

	18	**Display Pipe A Scan Line Wait Enable** 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** 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 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** 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** Source: RenderCS 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** 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.
	13	**Display Plane B Syncronous Flip Pending Wait Enable** 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** 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.

SYNC_FLIP_STATUS - Wait For Event and Display Flip Flags Register

11 **Reserved**

Format:	MBZ

10 **Display Pipe B Scan Line Wait Enable**

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**

Format:	Enable

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

8 **Display Pipe B H Blank Wait Enable**

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**

Format:	MBZ

4:0 **Condition Code Wait Select**

This field enables a wait for the duration that the corresponding condition code is active. These enable select one of 15 condition codes in the EXCC register, that cause the parser to wait until that condition-code in the EXCC is cleared.

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

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

1.1.14.7 SYNC_FLIP_STATUS_1 – Wait for Event and Display Flip Flags Register 1

SYNC_FLIP_STATUS_1 - Wait For Event and Display Flip Flags Register 1

Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32

Address:	022D4h
Name:	RCS Wait For Event and Display Flip Flags Register 1
ShortName:	RCS_SYNC_FLIP_STATUS_1

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.

DWord	Bit	Description
0	31:27	**Reserved** Format: MBZ
	26:15	**Reserved** Format: MBZ
	11	**SyncFlush Status** Format: Enable This field toggles on completion of sync flush. This bit toggle generates Interrupt and also reports interrupt status to HWSP on sync flush done.
	10	**Display Plane C Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane C Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	9	**Display Plane C Syncronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane C Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	8	**Display Sprite C Syncronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Sprite C Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.

SYNC_FLIP_STATUS_1 - Wait For Event and Display Flip Flags Register 1

	7	**Display Plane C Asyncronous Performance Flip Pending Wait Enable** Source: RenderCS Format: Enable This field enables a wait for the duration of a Display Plane C Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	6	**Display Plane C Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane C "Flip Pending" condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	5	**Display Plane C Syncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane C Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	4	**Display Sprite C Syncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Sprite C Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	3	**Reserved** Format: MBZ
	2	**Display Pipe C Scan Line Wait Enable** Format: Enable This field enables a wait while a Display Pipe C Scan Line condition exists. This condition is defined as the the start of the scan line specified in the Pipe C Display Scan Line Count Range Compare Register. See Scan Line Event in the Device Programming Interface chapter of MI Functions.
	1	**Display Pipe C Vertical Blank Wait Enable** Format: Enable This field enables a wait until the next Display Pipe C Vertical Blank event occurs. This event is defined as the start of the next Display Pipe C vertical blank period. Note that this can cause a wait for up to an entire refresh period. See Vertical Blank Event (See Programming Interface).
	0	**Display Pipe C H Blank Wait Enable** Format: Enable This field enables a wait until the start of next Display Pipe C Horizontal Blank event occurs. This event is defined as the start of the next Display C Horizontal blank period. Note that this can cause a wait for

SYNC_FLIP_STATUS_1 - Wait For Event and Display Flip Flags Register 1

		up to a line. See Horizontal Blank Event in the Device Programming Interface chapter of MI Functions.

1.1.15 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.

1.1.15.1 IA_VERTICES_COUNT — Reported Vertices Counter

IA_VERTICES_COUNT

Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000, 0x00000000
Access:	R/W
Size (in bits):	64
Trusted Type:	1
Address:	02310h

This register stores the count of vertices processed by VF. This register is part of the context save and restore.

DWord	Bit	Description
0	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.)

1.1.15.2 IA_PRIMITIVES_COUNT — Reported Vertex Fetch Output Primitives Counter

IA_PRIMITIVES_COUNT

Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000, 0x00000000
Access:	R/W
Size (in bits):	64
Trusted Type:	1
Address:	02318h

This register stores the count of primitives generated by VF. This register is part of the context save and restore.

DWord	Bit	Description
0	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.)

1.1.15.3 VS_INVOCATION_COUNT— Reported Vertex Shader Invocation Counter

VS_INVOCATION_COUNT

Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000, 0x00000000
Access:	R/W
Size (in bits):	64
Trusted Type:	1
Address:	02320h

This register stores the value of the vertex count shaded by VS. This register is part of the context save and restore.

DWord	Bit	Description
0	63:0	**VS Invocation Count Report** Number of vertices that are dispatched as threads by the VS stage. Updated only when **Statistics Enable** is set in VS_STATE (see the Vertex Shader Chapter in the 3D Volume.)

1.1.15.4 HS_INVOCATION_COUNT— Reported Hull Shader Invocation Counter

HS_INVOCATION_COUNT	
Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000, 0x00000000
Access:	R/W
Size (in bits):	64
Trusted Type:	1
Address:	02300h

This register stores the number of patch objects processed by the HS unit. E.g., A PATCHLIST_2 topology with 6 vertices would cause this counter to increment by 3 (there are 3 2-vertex patch objects in that topology).This register is part of the context save and restore.

DWord	Bit	Description
0	63:0	**HS Invocation Count** Number of patch objects processed by the HS stage. Updated only when HS Enable and HS Statistics Enable are set in 3DSTATE_HS

1.1.15.5 DS_INVOCATION_COUNT— Reported Domain Shader Invocation Counter

DS_INVOCATION_COUNT	
Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000, 0x00000000
Access:	R/W
Size (in bits):	64
Trusted Type:	1
Address:	02308h

This register stores the number of domain points shaded by the DS threads. Domain points which hit in the DS cache will not cause this register to increment. Note that the spawning of a DS thread which shades two domain points will cause this counter to increment by two.This register is part of the context save and restore.

DWord	Bit	Description
0	63:0	**DS Invocation Count** Number of domain points shaded by the DS threads. Updated only when DS Function Enable and Statistics Enable are set in 3DSTATE_DS

1.1.15.6 GS_INVOCATION_COUNT — Reported Geometry Shader Thread Invocation Counter

GS_INVOCATION_COUNT

Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000, 0x00000000
Access:	R/W
Size (in bits):	64
Trusted Type:	1
Address:	02328h

This register stores the number of objects that are part of geometry shader threads. This register is part of the context save and restore.

DWord	Bit	Description
0	63:0	**GS Invocation Count** Number of objects that are dispatched as a 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.)

1.1.15.7 GS_PRIMITIVES_COUNT — Reported Geometry Shader Output Primitives Counter

GS_PRIMITIVES_COUNT

Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000, 0x00000000
Access:	R/W
Size (in bits):	64
Trusted Type:	1
Address:	02330h

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.

DWord	Bit	Description
0	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.)

1.1.15.8 CL_INVOCATION_COUNT— Reported Clipper Thread Invocation Counter

CL_INVOCATION_COUNT	
Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000, 0x00000000
Access:	R/W
Size (in bits):	64
Trusted Type:	1
Address:	02338h

This register stores the count of objects entering the Clipper stage. This register is part of the context save and restore.

DWord	Bit	Description
0	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.)

1.1.15.9 CL_PRIMITIVES_COUNT— Reported Clipper Output Primitives Counter

CL_PRIMITIVES_COUNT	
Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000, 0x00000000
Access:	R/W
Size (in bits):	64
Trusted Type:	1
Address:	02340h

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.

DWord	Bit	Description
0	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.)

1.1.15.10 PS_INVOCATION_COUNT— Reported Pixels Shaded Counter

PS_INVOCATION_COUNT	
Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000, 0x00000000
Access:	R/W
Size (in bits):	64
Trusted Type:	1
Address:	02348h

DWord	Bit	Description
0	63:0	**PS Invocation Count** Reflects a count of the total number of pixels (including unlit “helper pixels” within a subspan that need to go through the PS shader to provide 2x2 gradients) 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.

1.1.15.11 PS_DEPTH_COUNT — Reported Pixels Passing Depth Test Counter

PS_DEPTH_COUNT	
Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000, 0x00000000
Access:	R/W
Size (in bits):	64
Trusted Type:	1
Address:	02350h

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.

DWord	Bit	Description
0	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.

1.1.15.12 TIMESTAMP — Reported Timestamp Count

TIMESTAMP - Reported Timestamp Count	
Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000, 0x00000000
Access:	RO. This register is not set by the context restore.
Size (in bits):	64
Address:	02358h

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.

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

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

1.1.15.13 SO_NUM_PRIMS_WRITTEN[0:3]— Stream Output Num Primitives Written Counters

SO_NUM_PRIMS_WRITTEN[0:3] - Stream Output Num Primitives Written Counter	
Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000, 0x00000000
Access:	R/W
Size (in bits):	64
Address:	05200h-0521Fh

There is one 64-bit register for each of the 4 supported streams:5200h-5207h SO_NUM_PRIMS_WRITTEN0 (for Stream Out Stream #0)5208h-520Fh SO_NUM_PRIMS_WRITTEN1 (for Stream Out Stream #1)5210h-5217h SO_NUM_PRIMS_WRITTEN2 (for Stream Out Stream #2)5218h-521Fh SO_NUM_PRIMS_WRITTEN3 (for Stream Out Stream #3)These registers are used to count the number of primitives (aka objects: points, lines, triangles) which the SO stage has successfully written to a particular “stream’s” Streamed Vertex Output buffers, subject to buffer overflow detection. (See the Stream Output section of the 3D pipeline volume).These registers are part of the context save and restore.

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

1.1.15.14 SO_PRIM_STORAGE_NEEDED[0:3] —Stream Output Primitive Storage Needed Counters

SO_PRIM_STORAGE_NEEDED[0:3] - Stream Output Primitive Storage Needed Counters

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000, 0x00000000
Access:	RO. This register is set by the context restore.
Size (in bits):	64
Address:	05240h-0525Fh

There is one 64-bit register for each of the 4 supported streams:

5240h-5247h SO_PRIM_STORAGE_NEEDED0 (for Stream Out Stream #0)
5248h-524Fh SO_PRIM_STORAGE_NEEDED1 (for Stream Out Stream #1)
5250h-5257h SO_PRIM_STORAGE_NEEDED2 (for Stream Out Stream #2)
5258h-525Fh SO_PRIM_STORAGE_NEEDED3 (for Stream Out Stream #3)

These registers are used to count the number of primitives (aka objects: points, lines, triangles) which the SO stage has or would have written to a particular “stream’s” Streamed Vertex Output buffers if all buffers had been large enough to accommodate the writes. (See the Stream Output section of the 3D pipeline volume).

These registers are part of the context save and restore.

DWord	Bit	Description
0	63:0	**Prim Storage Needed Count** Format: U64 This count is incremented (by one) by the SOL stage for each object (point, line, triangle) it writes or attempts to write to the corresponding stream’s output buffers. The count is not affected by the actual number of buffers bound to the stream.

1.1.15.15 SO_WRITE_OFFSET[0:3] —Stream Output Write Offsets

SO_WRITE_OFFSET[0:3] - Stream Output Write Offsets

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	RW. This register is set by the context restore.
Size (in bits):	32
Address:	05280h-0528Fh

There is one R/W 32-bit register for each of the 4 supported stream output buffer slots:

5280h-5283h SO_WRITE_OFFSET0 (for Stream Out Buffer #0)

5284h-5287h SO_ WRITE_OFFSET1 (for Stream Out Buffer #1)

5288h-528Bh SO_ WRITE_OFFSET2 (for Stream Out Buffer #2)

528Ch-528Fh SO_ WRITE_OFFSET3 (for Stream Out Buffer #3)

These registers are used to set and track a DWord-granular Write Offset for each of the 4 Stream Output Buffer slots. Software can directly write them via MI_LOAD_REGxxx commands. The SOL stage will increment them as part of stream output processing. Software can cause them to be written to memory via MI_STORE_REGxxx commands. (See the Stream Output section of the 3D pipeline volume).

These registers are part of the context save and restore.

Programming Notes

- Software must ensure that no HW stream output operations can be in process or otherwise pending at the point that the MI_LOAD/STORE commands are processed. This will likely require a pipeline flush.
- The SOL stage will effectively advance the write offset by the buffer's Surface Pitch after each vertex is written (assuming no overflow is detected in any targetted SO buffer). Under "normal" conditions one would expect software to initialize the WriteOffset to some (possibly zero) multiple of Surface Pitch in order to align vertex writes to the buffer's Base Address, though it is not required to do so.

DWord	Bit	Description
0	31:2	**Write Offset** Format: U30 This field contains a DWord offset from the corresponding SO buffer's Base Address value. The SOL stage uses this value as a write offset when performing writes to the buffer. The SOL stage will increment this value as a part of performing stream output to the buffer. Note that the SOL stage uses the buffer's Surface Pitch to advance the Write Offset, without regard to the buffer's Base Address (see Programming Notes above).
	1:0	**Reserved** Format: MBZ

1.1.16 Predicate Render Registers

1.1.16.1 MI_PREDICATE_SRC0 - Predicate Rendering Temporary Register0

MI_PREDICATE_SRC0 - Predicate Rendering Temporary Register0	
Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000, 0x00000000
Access:	R/W
Size (in bits):	64
Address:	02400h-02407h

DWord	Bit	Description
0	63:0	**MI_PREDICATE_SRC0** This register is a temporary register for Predicate Rendering. See Predicate Rendering section for more details.

1.1.16.2 1.1.16.2 MI_PREDICATE_SRC1– Predicate Rendering Temporary Register1

MI_PREDICATE_SRC1 - Predicate Rendering Temporary Register1	
Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000, 0x00000000
Access:	R/W
Address:	02408h-0240Fh

DWord	Bit	Description
0	63:0	**MI_PREDICATE_SRC1** This register is a temporary register for Predicate Rendering. See Predicate Rendering section for more details.

1.1.16.3 MI_PREDICATE_DATA– Predicate Rendering Data Storage

MI_PREDICATE_DATA - Predicate Rendering Data Storage

Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000, 0x00000000
Access:	R/W
Size (in bits):	64
Address:	02410h-02417h

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

1.1.16.4 MI_PREDICATE_RESULT – Predicate Rendering Data Result

MI_PREDICATE_RESULT - Predicate Rendering Data Result

Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Address:	02418h

DWord	Bit	Description
0	31:1	**Reserved** Format: MBZ
	0	**MI_PREDICATE_RESULT** This bit is the result of the last MI_PREDICATE.

1.1.17 AUTO_DRAW Registers

1.1.17.1 3DPRIM_END_OFFSET – Auto Draw End Offset

3DPRIM_END_OFFSET - Auto Draw End Offset	
Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Address:	02420h-02423h

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

1.1.17.2 3DPRIM_START_VERTEX – Load Indirect Start Vertex

3DPRIM_START_VERTEX - Load Indirect Start Vertex	
Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Address:	02430h-02433h

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

1.1.17.3 3DPRIM_VERTEX_COUNT – Load Indirect Vertex Count

3DPRIM_VERTEX_COUNT - Load Indirect Vertex Count

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Address:	02434h-02437h

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

1.1.17.4 3DPRIM_INSTANCE_COUNT – Load Indirect Instance Count

3DPRIM_INSTANCE_COUNT - Load Indirect Instance Count

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Address:	02438h-0243Bh

DWord	Bit	Description
0	31:0	**Instance Count** This register is used to store the Instance Count of the 3D_PRIMITIVE command when Load Indirect Enable is set.

1.1.17.5 3DPRIM_START_INSTANCE – Load Indirect Start Instance

3DPRIM_START_INSTANCE - Load Indirect Start Instance	
Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Address:	0243Ch-0243Fh

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

1.1.17.6 3DPRIM_BASE_VERTEX – Load Indirect Base Vertex

3DPRIM_BASE_VERTEX - Load Indirect Base Vertex	
Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Address:	02440h-02443h

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

1.1.18 MMIO Registers for GPGPU Indirect Dispatch

This register is normally written with the MI_LOAD_REGISTER_MEMORY command rather than from the CPU.

These registers should not be written with 0 for these projects. To avoid this, the MI_LOAD_REGISTER_MEMORY command which writes them from an address in memory which was written by a previous GPGPU_WALKER command will need to be checked with the following command sequence. The commands in red are the additional commands to implement the workaround:

MI_LOAD_REGISTER_MEMORY Xaddress, GPGPU_DISPATCHDIMX

MI_CONDITIONAL_BATCH_BUFFER_END Xaddress, 0 // Compare X dimension to 0, end batch buffer if 0

MI_LOAD_REGISTER_MEMORY GPGPU_DISPATCHDIMY

MI_CONDITIONAL_BATCH_BUFFER_END Yaddress, 0 // Compare Y dimension to 0, end batch buffer if 0

MI_LOAD_REGISTER_MEMORY GPGPU_DISPATCHDIMZ

MI_CONDITIONAL_BATCH_BUFFER_END Zaddress, 0 // Compare Z dimension to 0, end batch buffer if 0

GPGPU_WALKER // Walker with indirect dispatch

This way, if any dimension is 0 we would not execute the GPGPU_WALKER. This has the limitation that the indirect GPGPU_WALKER has to be the last WALKER of the batch buffer.

1.1.18.1 GPGPU_DISPATCHDIM(X/Y/Z) - GPGPU Dispatch Dimension (X/Y/Z)

These registers are normally written with the MI_LOAD_REGISTER_MEMORY command rather than from the CPU.

<table>
<tr><th colspan="3">GPGPU_DISPATCHDIMX - GPGPU Dispatch Dimension X</th></tr>
<tr><td colspan="2">Register Space:</td><td>MMIO: 0/2/0</td></tr>
<tr><td colspan="2">Source:</td><td>RenderCS</td></tr>
<tr><td colspan="2">Default Value:</td><td>0x00000000</td></tr>
<tr><td colspan="2">Access:</td><td>R/W</td></tr>
<tr><td colspan="2">Size (in bits):</td><td>32</td></tr>
<tr><td colspan="2">Address:</td><td>02500h</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:0</td><td><b>Dispatch Dimension X</b>
<table>
<tr><td>Format:</td><td>U32</td></tr>
</table>
The number of thread groups to be dispatched in the X dimension (max x + 1).
<table>
<tr><th>Value</th><th>Name</th><th>Project</th></tr>
<tr><td>1,FFFFFFFFh</td><td></td><td></td></tr>
</table>
</td></tr>
</table>

GPGPU_DISPATCHDIMY - GPGPU Dispatch Dimension Y

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Address:	02504h

DWord	Bit	Description		
0	31:0	**Dispatch Dimension Y**		
		Format:	U32	
		The number of thread groups to be dispatched in the Y dimension (max y + 1		
		Value	**Name**	**Project**
		1,FFFFFFFFh		

GPGPU_DISPATCHDIMZ - GPGPU Dispatch Dimension Z

Register Space:	MMIO: 0/2/0
Source:	RenderCS
Default Value:	0x00000000
Access:	R/W
Size (in bits):	32
Address:	02508h

DWord	Bit	Description		
0	31:0	**Dispatch Dimension Z**		
		Format:	U32	
		The number of thread groups to be dispatched in the Zdimension (max Z + 1)		
		Value	**Name**	**Project**
		1,FFFFFFFFh		

1.1.18.2 TS_GPGPU_THREADS_DISPATCHED – Count Active Channels Dispatched

TS_GPGPU_THREADS_DISPATCHED - Count Active Channels Dispatched	
Register Space:	MMIO: 0/2/0
Project:	All
Source:	RenderCS
Default Value:	0x00000000, 0x00000000
Access:	R/W
Size (in bits):	64
Trusted Type:	1
Address:	02290h

This register is used to count the number of active channels that TS sends for dispatch. For each dispatch the active bits in the execution mask are summed and added to this register. This register is reset when a write occurs to 2290h

DWord	Bit	Description
0	63:0	**GPGPU_THREADS_DISPATCHED** Format: U64 This count is increased by the number of active bits in the execution mask each time the TS sends a GPGPU dispatch.

1.1.19 Memory Interface Registers

1.1.19.1 PWRCTX_REST_DONE - Power Context Restore Done

PWRCTX_REST_DONE - Power Context Restore Done	
Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	04000h-04003h

Power Context Restore Done

DWord	Bit	Description
0	31:16	**Mask Bits** Default Value: 0000h Access: RO Mask Bits
	15:1	**Extra Mask Bits** Default Value: 000000000000000b Access: R/W

PWRCTX_REST_DONE - Power Context Restore Done

		Extra Mask Bits
	0	**Restore Done** Default Value: 0b Access: R/W GAM - CS will write to indicate 'restore done' It is a config message register between CS & GAM

1.1.19.2 WR_WATERMARK - Write Watermark

WR_WATERMARK - Write Watermark

Register Space:	MMIO: 0/2/0
Default Value:	0x000FFEA4
Address:	04028h-0402Bh

Write Watermark

DWord	Bit	Description
0	31:20	**Counter Extra Bits** Default Value: 000000000000b Access: R/W Counter Extra Bits
	19	**Watermark Timeout Enable** Default Value: 1b Access: R/W Watermark timeout enable.
	18:8	**Watermark Timeout** Default Value: 11111111110b Access: R/W Number of clocks that the write pipe queue is allowed to keep a ready write cycle, without reads or writes to the queue. Once this value is met, and if the feature is enabled, the watermark will be considered reach, and all pending write requests will be issued.

WR_WATERMARK - Write Watermark

	7	**Watermark En** Default Value: 1b Access: R/W Enable write request grouping
	6:0	**High Watermark** Default Value: 0100100b Access: R/W This is the number of write requests to be collected before initiating a write burst. Once a burst is initiated, it will continue until all the available writes are requested.

1.1.19.3 GFX_PRIO_CTRL - GFX Arbiter Client Priority Control

GFX_PRIO_CTRL - GFX Arbiter Client Priority Control

Register Space:	MMIO: 0/2/0
Default Value:	0x00011D10
Address:	0402Ch-0402Fh

GFX Arbiter Client Priority Control

DWord	Bit	Description
0	31:17	**Extra 402C Register** Default Value: 000000000000000b Access: R/W Extra 402C Register
	16:12	**Read Rstrm Max Reject** Default Value: 10001b Access: R/W Read Rstrm Max Reject
	11:9	**gapc_gam_c_priority** Default Value: 110b Access: R/W

GFX_PRIO_CTRL - GFX Arbiter Client Priority Control

		gapc_gam_c_priority - Lowest Bit [9] is not used
	8:6	**gapc_gam_z_priority** Default Value: 100b Access: R/W gapc_gam_z_priority - Lowest Bit [6] is not used
	5:3	**gapc_gam_l3_priority** Default Value: 010b Access: R/W gapc_gam_l3_priority - Lowest Bit [3] is not used
	2:0	**gafm_gam_priority** Default Value: 000b Access: R/W Client Priority control bitss gafm_gam_priority - Lowest Bit [0] is not used

1.1.19.4 1.1.19.4 GFX_PEND_TLB_0 - Max Outstanding Pending TLB Requests 0

GFX_PEND_TLB_0 - Max Outstanding Pending TLB Requests 0

Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	04034h-04037h

GFX_PEND_TLB_0 - Max Outstanding Pending TLB Requests 0

DWord	Bit	Description	
0	31	**TEX Limit Enable Bit**	
		Default Value:	0b

GFX_PEND_TLB_0 - Max Outstanding Pending TLB Requests 0

Access:	R/W

TEX Limit Enable bit Project: All Format: U1

This bit is used to enable the pending TLB requests limitation function for the Texture Cache

When set, the number of internal pending read requests which require a TLB read will not exceed the programmed counter value.

30 **Reserved Bit**

Default Value:	0b
Access:	RO

Reserved Project: All Format: MBZ

29:24 **TEX TLB Limit Count**

Default Value:	000000b
Access:	R/W

TEX TLB Limit Count Project: All Format: U6

This is the MAX number of Allowed internal pending read requests which require a TLB read

23 **DC Limit Enable bit**

Default Value:	0b
Access:	R/W

DC Limit Enable bit Project: All Format: U1

This bit is used to enable the pending TLB requests limitation function for the Instruction Cache.

When set, the number of internal pending read requests which require a TLB read will not exceed the programmed counter value.

22 **Reserved Bit**

Default Value:	0b
Access:	RO

Reserved Project: All Format: MBZ

21:16 **DC TLB Limit Count**

Default Value:	000000b

GFX_PEND_TLB_0 - Max Outstanding Pending TLB Requests 0

Access:	R/W

DC TLB Limit Count Project:All Format:U6

This is the MAX number of Allowed internal pending read requests which require a TLB read.

15 **VF Limit Enable bit**

Default Value:	0b
Access:	R/W

VF Limit Enable bit Project: All Format: U1

This bit is used to enable the pending TLB requests limitation function for the Vertex Fetch

When set, the number of internal pending read requests which require a TLB read will not exceed the programmed counter value.

14 **Reserved Bit**

Default Value:	0b
Access:	RO

Reserved Project: All Format: MBZ

13:8 **VF TLB Limit Count**

Default Value:	000000b
Access:	R/W

VF TLB Limit Count Project: All Format: U6

This is the MAX number of Allowed internal pending read requests which require a TLB read.

7 **VMC Limit Enable bit**

Default Value:	0b
Access:	R/W

VMC Limit Enable bit Project: All Format: U1

This bit is used to enable the pending TLB requests limitation function for the Video Motion Compensation . When set, the number of internal pending read requests which require a TLB read will not exceed the programmed counter value.

6 **Reserved Bit**

Default Value:	0b
Access:	RO

GFX_PEND_TLB_0 - Max Outstanding Pending TLB Requests 0

		Reserved Project: All Format: MBZ
	5:0	**VMC TLB Limit Count** Default Value: 000000b Access: R/W VMC TLB Limit Count Project: All Format: U6 This is the MAX number of Allowed internal pending read requests which require a TLB read.

B/D/F/Type:0/0/0/GAMunit_Config

Address Offset:4034-4037h

Default Value:00000000h

Access: RO; RW;

Size:32 bits

GFX_PEND_TLB_0 - Max Outstanding Pending TLB Requests 0

Bit	Access	Default Value	RST/PWR	Description
31	RW	0b	Core	**TEX Limit Enable Bit (TEXLEN):** **TEX Limit Enable bit** Project: All Format: U1 This bit is used to enable the pending TLB requests limitation function for the Texture Cache When set, the number of internal pending read requests which require a TLB read will not exceed the programmed counter value.
30	RO	0b	Core	**Reserved Bit (RSVD):** **Reserved** Project: All Format: MBZ
29:24	RW	000000b	Core	**TEX TLB Limit Count (TEXTLBLCNT):** **TEX TLB Limit Count** Project: All Format: U6 This is the MAX number of Allowed internal pending read requests which require a TLB read
23	RW	0b	Core	**DC Limit Enable bit (DCLEN):** **DC Limit Enable bit** Project: All Format: U1

Bit	Access	Default Value	RST/PWR	Description
				This bit is used to enable the pending TLB requests limitation function for the Instruction Cache. When set, the number of internal pending read requests which require a TLB read will not exceed the programmed counter value.
22	RO	0b	Core	**Reserved Bit (RSVD):** **Reserved** Project: All Format: MBZ
21:16	RW	000000b	Core	**DC TLB Limit Count (DCTLBLCNT):** **DC TLB Limit Count** Project:All Format:U6 This is the MAX number of Allowed internal pending read requests which require a TLB read.
15	RW	0b	Core	**VF Limit Enable bit (VFLEN):** **VF Limit Enable bit** Project: All Format: U1 This bit is used to enable the pending TLB requests limitation function for the Vertex Fetch When set, the number of internal pending read requests which require a TLB read will not exceed the programmed counter value.
14	RO	0b	Core	**Reserved Bit (RSVD):** **Reserved** Project: All Format: MBZ
13:8	RW	000000b	Core	**VF TLB Limit Count (VFTLBLCNT):** **VF TLB Limit Count** Project: All Format: U6 This is the MAX number of Allowed internal pending read requests which require a TLB read.
7	RW	0b	Core	**VMC Limit Enable bit (VMCLEN):** **VMC Limit Enable bit** Project: All Format: U1 This bit is used to enable the pending TLB requests limitation function for the Video Motion Compensation. When set, the number of internal pending read requests which require a TLB read will not exceed the programmed counter value.
6	RO	0b	Core	**Reserved Bit (RSVD):** **Reserved** Project: All Format: MBZ

Bit	Access	Default Value	RST/PWR	Description
5:0	RW	000000b	Core	**VMC TLB Limit Count (VMCTLBLCNT):** **VMC TLB Limit Count** Project: All Format: U6 This is the MAX number of Allowed internal pending read requests which require a TLB read.

1.1.19.5 GFX_PEND_TLB_1 - Max Outstanding Pending TLB Requests 1

GFX_PEND_TLB_1 - Max Outstanding Pending TLB Requests 1	
Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	04038h-0403Bh

GFX_PEND_TLB_1 - Max Outstanding pending TLB requests 1

DWord	Bit	Description
0	31	**SOL Limit Enable bit** Default Value: 0b Access: R/W SOL Limit Enable bit Project: All Format: U1 This bit is used to enable the pending TLB requests limitation function for the SOL. When set, the number of internal pending read requests which require a TLB read will not exceed the programmed counter value.
	30	**Reserved Bits** Default Value: 0b Access: RO Reserved Project: All Format: MBZ
	29:24	**SOL TLB Limit Count** Default Value: 000000b Access: R/W SOL TLB Limit Count Project: All Format: U6 This is the MAX number of Allowed internal pending read requests which require a TLB read.
	23	**L3 Limit Enable bit** Default Value: 0b Access: R/W

GFX_PEND_TLB_1 - Max Outstanding Pending TLB Requests 1

L3 Limit Enable bit Project: All Format: U1

This bit is used to enable the pending TLB requests limitation function for the L3. When set, the number of internal pending read requests which require a TLB read will not exceed the programmed counter value.

22 **Reserved Bit**

Default Value:	0b
Access:	RO

Reserved Project: All Format: MBZ

21:16 **L3 TLB Limit Count**

Default Value:	000000b
Access:	R/W

L3 TLB Limit Count Project: All Format: U6

This is the MAX number of Allowed internal pending read requests which require a TLB read.

15 **RCZ Limit Enable bit**

Default Value:	0b
Access:	R/W

RCZ Limit Enable bit Project: All Format: U1

This bit is used to enable the pending TLB requests limitation function for the Render Depth Cache

When set, the number of internal pending read requests which require a TLB read will not exceed the programmed counter value.

14 **Reserved Bit**

Default Value:	0b
Access:	RO

Reserved Project: All Format: MBZ

Programming Notes
""

13:8 **RCZ TLB Limit Count**

Default Value:	000000b
Access:	R/W

RCZ TLB Limit Count Project: All Format: U6

This is the MAX number of Allowed internal pending read requests which require a TLB read.

GFX_PEND_TLB_1 - Max Outstanding Pending TLB Requests 1

	7	**RCC Limit Enable bit** Default Value: 0b Access: R/W RCC Limit Enable bit Project: All Format: U1 This bit is used to enable the pending TLB requests limitation function for the Render Color Cache. When set, the number of internal pending read requests which require a TLB read will not exceed the programmed counter value.
	6	**Reserved Bit** Default Value: 0b Access: RO Reserved Project: All Format: MBZ
	5:0	**RCC TLB Limit Count** Default Value: 000000b Access: R/W RCC TLB Limit Count Project: All Format: U6 This is the MAX number of Allowed internal pending read requests which require a TLB read.

1.1.19.6 L3_LRA_0 - L3 LRA 0

L3_LRA_0 - L3 LRA 0

Register Space:	MMIO: 0/2/0
Default Value:	0x3F201F00
Address:	0403Ch-0403Fh

L3 LRA 0

DWord	Bit	Description
0	31:24	**L3 LRA1 Max** Default Value: 00111111b Access: R/W L3 LRA1 Max Project: All Format: U6 Maximum value of programmable LRA1

L3_LRA_0 - L3 LRA 0

	23:16	**L3 LRA1 Min** Default Value: 00100000b Access: R/W L3 LRA1 Min Project: All Format: U6 Minimum value of programmable LRA1
	15:8	**L3 LRA0 Max** Default Value: 00011111b Access: R/W L3 LRA0 Max Project: All Format: U6 Maximum value of programmable LRA0
	7:0	**L3 LRA0 Min** Default Value: 00000000b Access: R/W L3 LRA0 Min Project: All Format: U6 Minimum value of programmable LRA1

1.1.19.7 L3_LRA_1 - L3 LRA 1

L3_LRA_1 - L3 LRA 1

Register Space:	MMIO: 0/2/0
Default Value:	0x0900FF40
Address:	04040h-04043h

L3 LRA 1

DWord	Bit	Description
0	31:30	**Reserved Bits** Default Value: 00b Access: RO Reserved Bits
	29:28	**DC** Default Value: 00b Access: R/W Which LRA should DC use

L3_LRA_1 - L3 LRA 1

	27:26	**TEXTURE** Default Value: 10b Access: R/W Which LRA should TEXTURE use
	25:24	**L3** Default Value: 01b Access: R/W Which LRA should L3 use
	23:16	**Reserved Bits** Default Value: 00000000b Access: RO Reserved Bits
	15:8	**L3 LRA2 Max** Default Value: 11111111b Access: R/W L3 LRA2 Max Project: All Format: U6 Maximum value of programmable LRA2
	7:0	**L3 LRA2 Min** Default Value: 01000000b Access: R/W L3 LRA2 Min Project: All Format: U6 Minimum value of programmable LRA2

1.1.19.8 CVS_TLB_LRA_0 - CVS TLB LRA 0

CVS_TLB_LRA_0 - CVS TLB LRA 0	
Register Space:	MMIO: 0/2/0
Default Value:	0x1F080700
Address:	04044h-04047h

CVS TLB LRA 0

DWord	Bit	Description
0	31:29	**Reserved Bits** Default Value: 000b Access: RO Reserved Project: All Format: MBZ
	28:24	**CVS LRA1 Max** Default Value: 11111b Access: R/W CVS LRA1 Max Project: All Format: MBZ Maximum value of programmable LRA1
	23:21	**Reserved Bits** Default Value: 000b Access: RO Reserved Project: All Format: MBZ
	20:16	**CVS LRA1 Min** Default Value: 01000b Access: R/W CVS LRA1 Min Project: All Format: U6 Minimum value of programmable LRA1
	15:13	**Reserved Bits** Default Value: 000b Access: RO Reserved Project: All Format: MBZ
	12:8	**CVS LRA0 Max** Default Value: 00111b Access: R/W CVS LRA0 Max Project: All Format: MBZ Maximum value of programmable LRA0

CVS_TLB_LRA_0 - CVS TLB LRA 0

	7:5	**Reserved Bits** Default Value: 000b Access: RO Reserved Project: All Format: MBZ
	4:0	**CVS LRA0 Min** Default Value: 00000b Access: R/W CVS LRA0 Min Project: All Format: U6 Minimum value of programmable LRA0

1.1.19.9 _TLB_LRA_1 - CVS TLB LRA 1

CVS_TLB_LRA_1 - CVS TLB LRA 1

Register Space:	MMIO: 0/2/0
Default Value:	0x00001F18
Address:	04048h-0404Bh

CVS TLB LRA 1

DWord	Bit	Description
0	31:13	**Reserved Bits** Default Value: 0000000000000000000b Access: RO Reserved Project: All Format: MBZ
	12:8	**CVS LRA2 Max** Default Value: 11111b Access: R/W CVS LRA2 Max Project: All Format: MBZ Maximum value of programmable LRA2
	7:5	**Reserved Bits** Default Value: 000b Access: RO Reserved Project: All Format: MBZ
	4:0	**CVS LRA2 Min**

CVS_TLB_LRA_1 - CVS TLB LRA 1		
		Default Value: 11000b Access: R/W CVS LRA2 Min Project: All Format: U6 Minimum value of programmable LRA2

1.1.19.10 CVS_TLB_LRA_2 - CVS TLB LRA 2

CVS_TLB_LRA_2 - CVS TLB LRA 2	
Register Space:	MMIO: 0/2/0
Default Value:	0x00000005
Address:	0404Ch-0404Fh

CVS TLB LRA 2

DWord	Bit	Description
0	31:6	**Reserved Bits** Default Value: 00000000000000000000000000b Access: RO Reserved Project: All Format: MBZ
	5:4	**CS LRA** Default Value: 00b Access: R/W CS LRA Project: All Format: U6 Which LRA should CS use
	3:2	**VF LRA** Default Value: 01b Access: R/W VF LRA Project: All Format: U1 Which LRA should VF use
	1:0	**SO LRA** Default Value: 01b Access: R/W SO LRA Project: All Format: MBZ Which LRA should SO use

1.1.19.11 ZTLB_LRA_0 - ZTLB LRA 0

ZTLB_LRA_0 - ZTLB LRA 0	
Register Space:	MMIO: 0/2/0
Default Value:	0x1F107F00
Address:	04050h-04053h
ZTLB TLB LRA 0	

DWord	Bit	Description
0	31	**Reserved Bits** Default Value: 0b Access: RO Reserved Bits
	30:24	**ZTLB LRA1 Max** Default Value: 0011111b Access: R/W ZTLB LRA1 Max Project: All Format: U6 Maximum value of programmable LRA1
	23	**Reserved Bit** Default Value: 0b Access: RO Reserved Project: All Format: U1
	22:16	**ZTLB LRA1 Min** Default Value: 0010000b Access: R/W ZTLB LRA1 Min Project: All Format: MBZ Minimum value of programmable LRA1
	15	**Reserved Bits** Default Value: 0b Access: RO Reserved Bits
	14:8	**ZTLB LRA0 Max** Default Value: 1111111b Access: R/W ZTLB LRA0 Max Project: All Format: U1 Maximum value of programmable LRA0

ZTLB_LRA_0 - ZTLB LRA 0

	7	**Reserved Bit** Default Value: 0b Access: RO Reserved Project: All Format: U1
	6:0	**ZTLB LRA0 Min** Default Value: 0000000b Access: R/W ZTLB LRA0 Min Project: All Format: U6 Minimum value of programmable LRA0

1.1.19.12 ZTLB_LRA_1 - ZTLB LRA 1

ZTLB_LRA_1 - ZTLB LRA 1

Register Space:	MMIO: 0/2/0
Default Value:	0x00002F20
Address:	04054h-04057h

ZTLB TLB LRA 1

DWord	Bit	Description
0	31:22	**Reserved Bits** Default Value: 0000000000b Access: RO Reserved Project: All Format: MBZ
	21:20	**STC LRA** Default Value: 00b Access: R/W STC LRA Project: All Format: U6 Which LRA should STC use
	19:18	**HIZ LRA** Default Value: 00b Access: R/W HIZ LRA Project: All Format: U1 Which LRA should HIZ use

ZTLB_LRA_1 - ZTLB LRA 1

	17:16	**RCZ LRA** Default Value: 00b Access: R/W RCZ LRA Project: All Format: MBZ Which LRA should RCZ use
	15	**Reserved Bits** Default Value: 0b Access: RO Reserved Bits
	14:8	**ZTLB LRA2 Max** Default Value: 0101111b Access: R/W ZTLB LRA2 Max Project: All Format: U1 Maximum value of programmable LRA2
	7	**Reserved Bits** Default Value: 0b Access: RO Reserved Project: All Format: MBZ
	6:0	**ZTLB LRA2 Min** Default Value: 0100000b Access: R/W ZTLB LRA2 Min Project: All Format: U6 Minimum value of programmable LRA2

1.1.19.13 RCC_LRA_0 - RCC LRA 0

RCC_LRA_0 - RCC LRA 0	
Register Space:	MMIO: 0/2/0
Default Value:	0x3F100F00
Address:	04058h-0405Bh
RCC LRA 0	

DWord	Bit	Description
0	31:30	**Reserved Bit** Default Value: 00b Access: RO Reserved Project: All Format: U1
	29:24	**RCC LRA1 Max** Default Value: 111111b Access: R/W RCC LRA1 Max Project: All Format: U6 Maximum value of programmable LRA1
	23:22	**Reserved Bit** Default Value: 00b Access: RO Reserved Project: All Format: U1
	21:16	**RCC LRA1 Min** Default Value: 010000b Access: R/W RCC LRA1 Min Project: All Format: MBZ Minimum value of programmable LRA1
	15:14	**Reserved Bit** Default Value: 00b Access: RO Reserved Project: All Format: U1
	13:8	**RCC LRA0 Max** Default Value: 001111b Access: R/W RCC LRA0 Max Project: All Format: U1 Maximum value of programmable LRA0

RCC_LRA_0 - RCC LRA 0

	7:6	**Reserved Bit** Default Value: 00b Access: RO Reserved Project: All Format: U1
	5:0	**RCC LRA0 Min** Default Value: 000000b Access: R/W RCC LRA0 Min Project: All Format: U6 Minimum value of programmable LRA0

1.1.19.14 RCC_LRA_1 - RCC LRA 1

RCC_LRA_1 - RCC LRA 1

Register Space:	MMIO: 0/2/0
Default Value:	0x00010000
Address:	0405Ch-0405Fh

RCC LRA 1

DWord	Bit	Description
0	31:20	**Reserved Bits** Default Value: 000000000000b Access: RO Reserved Project: All Format: MBZ
	19:18	**MSC LRA** Default Value: 00b Access: R/W MSC LRA Project: All Format: U1 Which LRA should MSC use
	17:16	**RCC LRA** Default Value: 01b Access: R/W RCC LRA Project: All Format: MBZ

RCC_LRA_1 - RCC LRA 1

		Which LRA should RCC use
	15:0	**Reserved Bits** Default Value: 0000000000000000b Access: RO Reserved Project: All Format: MBZ

1.1.19.15 CASC_LRA_0 - CASC LRA 0

CASC_LRA_0 - CASC LRA 0

Register Space:	MMIO: 0/2/0
Default Value:	0x1F100F00
Address:	04060h-04063h

CASC LRA 0

DWord	Bit	Description
0	31:24	**CASC LRA1 Max** Default Value: 00011111b Access: R/W CASC LRA1 Max Project: All Format: U6 Maximum value of programmable LRA1 Maximum Allow Value: 159
	23:16	**CASC LRA1 Min** Default Value: 00010000b Access: R/W CASC LRA1 Min Project: All Format: U6 Minimum value of programmable LRA1
	15:8	**CASC LRA0 Max** Default Value: 00001111b Access: R/W CASC LRA0 Max Project: All Format: U6 Maximum value of programmable LRA0

CASC_LRA_0 - CASC LRA 0		
		Maximum Allow Value: 159
	7:0	**CASC LRA0 Min** Default Value: 00000000b Access: R/W CASC LRA0 Min Project: All Format: U6 Minimum value of programmable LRA1

1.1.19.16 CASC_LRA_1 - CASC LRA 1

CASC_LRA_1 - CASC LRA 1		
Register Space:		MMIO: 0/2/0
Default Value:		0x3F302F20
Address:	04064h-04067h	
CASC LRA 1		
DWord	**Bit**	**Description**
0	31:24	**CASC LRA3 Max** Default Value: 00111111b Access: R/W CASC LRA3 Max Project: All Format: U6 Maximum value of programmable LRA3
	23:16	**CASC LRA3 Min** Default Value: 00110000b Access: R/W CASC LRA3 Min Project: All Format: U6 Minimum value of programmable LRA3
	15:8	**CASC LRA2 Max** Default Value: 00101111b Access: R/W CASC LRA2 Max Project: All Format: U6 Maximum value of programmable LRA2
	7:0	**CASC LRA2 Min** Default Value: 00100000b

CASC_LRA_1 - CASC LRA 1		
		Access: R/W CASC LRA2 Min Project: All Format: U6 Minimum value of programmable LRA2

1.1.19.17 CASC_LRA_2 - CASC LRA 2

CASC_LRA_2 - CASC LRA 2

Register Space:	MMIO: 0/2/0
Default Value:	0x00009F40
Address:	04068h-0406Bh

CASC LRA 2

DWord	Bit	Description
0	31:16	**Reserved Bits** Default Value: 0000h Access: RO Reserved Project: All Format: MBZ
	15:8	**CASC LRA4 Max** Default Value: 10011111b Access: R/W CASC LRA4 Max Project: All Format: U6 Maximum value of programmable LRA4 Maximum Allow Value: 159
	7:0	**CASC LRA4 Min** Default Value: 01000000b Access: R/W CASC LRA4 Min Project: All Format: U6 Minimum value of programmable LRA4

1.1.19.18 CASC_LRA_3 - CASC LRA 3

CASC_LRA_3 - CASC LRA 3		
Register Space:		MMIO: 0/2/0
Default Value:		0x000014E4
Address:	0406Ch-0406Fh	
CASC LRA 3		

DWord	Bit	Description
0	31:18	**Reserved Bits** Default Value: 00000000000000b Access: RO Reserved Project: All Format: MBZ
	17:15	**BCS LRA** Default Value: 000b Access: R/W BCS LRA Project: All Format: U6 Which LRA should use
	14:12	**BLB LRA** Default Value: 001b Access: R/W BLB LRA Project: All Format: U6 Which LRA should use
	11:9	**VCS LRA** Default Value: 010b Access: R/W VCS LRA Project: All Format: U6 Which LRA should use
	8:6	**VMX LRA** Default Value: 011b Access: R/W VMX LRA Project: All Format: U6 Which LRA should use
	5:3	**VMC LRA** Default Value: 100b Access: R/W

CASC_LRA_3 - CASC LRA 3		
		VMC LRA Project: All Format: U6 Which LRA should use
	2:0	**VCR LRA** Default Value: 100b Access: R/W VCR LRA Project: All Format: U6 Which LRA should use

1.1.19.19 MEDIA_MAX_REQ_COUNT - MAX Requests Allowed - CASC

MEDIA_MAX_REQ_COUNT - MAX Requests Allowed - CASC	
Register Space:	MMIO: 0/2/0
Default Value:	0x10202020
Address:	04070h-04073h

Programmable Request Count - CASC

DWord	Bit	Description
0	31:24	**GFX Max Request Limit Count** Default Value: 00010000b Access: R/W This is the MAX number of Allowed Requests Count - These counters keep track of the accepted requests from each engine . Requests are counted, regardless of kind of cycle (Miss/Hit/Present) Minimum count value must be = 1
	23:16	**MFX/BLT Max Request Limit Count** Default Value: 00100000b Access: R/W This is the MAX number of Allowed Requests Count - These counters keep track of the accepted requests from each engine . Requests are counted, regardless of kind of cycle (Miss/Hit/Present) Minimum count value must be = 1
	15:14	**Reserved Bits** Default Value: 00b

MEDIA_MAX_REQ_COUNT - MAX Requests Allowed - CASC

DWord	Bit	Description
		Access: RO Reserved Bits
	13:8	**VLF Max Request Limit Count** Default Value: 100000b Access: R/W This is the MAX number of Allowed Requests Count - These counters keep track of the accepted requests from each client. Requests are counted, regardless of kind of cycle (Miss/Hit/Present) Minimum count value must be = 1
	7:6	**Reserved Bits** Default Value: 00b Access: RO Reserved Project: All Format: MBZ
	5:0	**CASC Max Request Limit Count** Default Value: 100000b Access: R/W This is the MAX number of Allowed Requests Count - These counters keep track of the accepted requests from each client. Requests are counted, regardless of kind of cycle (Miss/Hit/Present) Minimum count value must be = 1

1.1.19.20 GFX_MAX_REQ_COUNT - MAX Requests Allowed - GAM

GFX_MAX_REQ_COUNT - MAX Requests Allowed - GAM

Register Space:	MMIO: 0/2/0
Default Value:	0x43F20101
Address:	04074h-04077h

Programmable Request Count - GAM

DWord	Bit	Description
0	31:26	**GAP Writes Max Request Limit Count** Default Value: 010000b Access: R/W This is the MAX number of Allowed Write Requests Count - These counters keep track of the accepted write requests from all GAP clients (RCZ, HiZ,Stc, RCC, L3). Minimum count value must be = 1

GFX_MAX_REQ_COUNT - MAX Requests Allowed - GAM

25:20 **CVS Max Request Limit Count**

Default Value:	111111b
Access:	R/W

This is the MAX number of Allowed Requests Count - These counters keep track of the accepted requests from each client. Requests are counted, regardless of kind of cycle (Miss/Hit/Present)

Minimum count value must be = 1

19 **Reserved Bits**

Default Value:	0b
Access:	RO

Reserved Project: All Format: MBZ

18:13 **L3 Max Request Limit Count**

Default Value:	010000b
Access:	R/W

This is the MAX number of Allowed Requests Count - These counters keep track of the accepted requests from each client. Requests are counted, regardless of kind of cycle (Miss/Hit/Present)

Minimum count value must be = 1

12 **Reserved Bits**

Default Value:	0b
Access:	RO

Reserved Project: All Format: MBZ

11:6 **Z Request Limit Count**

Default Value:	000100b
Access:	R/W

This is the MAX number of Allowed Requests Count - These counters keep track of the accepted requests from each client. Requests are counted, regardless of kind of cycle (Miss/Hit/Present)

Minimum count value must be = 1

5:0 **RCC Request Limit Count**

Default Value:	000001b
Access:	R/W

This is the MAX number of Allowed Requests Count - These counters keep track of the accepted requests from each client. Requests are counted, regardless of kind of cycle (Miss/Hit/Present)

Minimum count value must be = 1

1.1.19.21 GAM_HWSP_REG - GAM Hardware Status Page Address Register

GAM_HWSP_REG - GAM Hardware Status Page Address Register

Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	04080h-04083h

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. 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.

DWord	Bit	Description
0	31:12	**GAM HWSP Register** Default Value: 00000h Access: R/W
	11:0	**Reserved Bits** Default Value: 000h Access: RO

1.1.19.22 GFX_ENG_FR - Graphics Engine Fault Register

GFX_ENG_FR - Graphics Engine Fault Register

Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	04094h-04097h

Graphics Engine Fault Register

DWord	Bit	Description
0	31:12	**Virtual Address of Fault** Default Value: 00000h Access: R/W This is the original Address of the Page that generated the First fault for this engine. This value is locked and not updated on subsequent faults, until the valid bit of this register is cleared by SW

GFX_ENG_FR - Graphics Engine Fault Register

	11	**GTTSEL** Default Value: 0b Access: R/W This bit indicates if the valid bit happened while using PPGTT or GGTT: 0 - PPGTT, 1 - GGTT This value is locked and not updated on subsequent faults, until the valid bit of this register is cleared by SW
	10:3	**SRCID of Fault** Default Value: 00h Access: R/W This is the Source ID of the unit that requested the cycle that generated the First Page fault for this engine. This value is locked and not updated on subsequent faults, until the valid bit of this register is cleared by SW
	2:1	**Fault Type** Default Value: 00b Access: R/W Type of Fault recorded: 00 - Page Fault. 01 - Invalid PD Fault 10 - Unloaded PD Fault 11 - Invalid and Unloaded PD fault This value is locked and not updated on subsequent faults, until the valid bit of this register is cleared by SW
	0	**Valid Bit** Default Value: 0b Access: R/W This bit indicates that the first fault for this engine has been recorded. It can only be cleared by SW, which will also clear the other fields.

1.1.19.23 ERROR - Main Graphic Arbiter Error Report

ERROR - Main Graphic Arbiter Error Report

Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	040A0h-040A3h

This register is used to report differet error conditions. Error bits are writable.

DWord	Bit	Description
0	31:16	**Reserved Bits** Default Value: 0000h Access: RO Reserved Bits
	15	**Reserved Error Bits 15** Default Value: 0b Access: R/W Reserved Error bits (Future expansion)
	14	**Reserved Error Bits 14** Default Value: 0b Access: R/W Reserved Error bits (Future expansion)
	13	**Reserved Error Bits 13** Default Value: 0b Access: R/W Reserved Error bits (Future expansion)
	12	**Reserved Error Bits 12** Default Value: 0b Access: R/W Reserved Error bits (Future expansion)
	11	**Reserved Error Bits 11** Default Value: 0b Access: R/W Reserved Error bits (Future expansion)
	10	**Reserved Error Bits 10**

ERROR - Main Graphic Arbiter Error Report

		Default Value:	0b
		Access:	R/W
		Reserved Error bits (Future expansion)	
	9	**Reserved Error Bits 9**	
		Default Value:	0b
		Access:	R/W
		Reserved Error bits (Future expansion)	
	8	**Unloaded PD Error**	
		Default Value:	0b
		Access:	R/W
		Unloaded PD error The Cache Line containing a PD entry being accessed, was marked as invalid in the last PD load cycle.	
	7	**Reserved Error Bits 7**	
		Default Value:	0b
		Access:	R/W
		Reserved Error bits (Future expansion)	
	6	**Page Directory Entry VTD Translation Error**	
		Default Value:	0b
		Access:	R/W
		Page Directory entry VTD translation error PD entry’s VTD translation generated an error (HPA is not accessible for DMA read or write)	
	4	**TLB Page VTD Translation Error**	
		Default Value:	0b
		Access:	R/W
		TLB Page VTD translation error A TLB Page’s VTD translation generated an error (HPA is not accessible for DMA read or write)	

ERROR - Main Graphic Arbiter Error Report

	2	**Invalid Page Directory Entry Error** Default Value: 0b Access: R/W Invalid Page Directory entry error PD entry’s valid bit is 0
	0	**TLB Page Fault Error** Default Value: 0b Access: R/W TLB Page Fault error A TLB Page’s GTT translation generated a page fault (GTT entry not valid)

1.1.19.24 DONE_REG - Gam Fub Done Lookup Register

DONE_REG - Gam Fub Done Lookup Register

Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	040B0h-040B3h

Gam Fub Done Lookup Register

DWord	Bit	Description
0	31:0	**Gam Fub Done Lookup Reg** Default Value: 00000000h Access: RO 31 CVS Credit Fifo is Empty 30 CVS TLB Don’t have any Cycles 29 Z Credit fifo is empty 28 ZTLB Don’t have any cycles 27 RCC Credit Fifo is empty 26 RCC TLB Don’t have any cycles 25 L3 Credit fifo is empty

DONE_REG - Gam Fub Done Lookup Register

		24	L3 TLB is don’t have any Cycles
		23	VLF Credit fifo is empty
		22	VLF TLB don’t have any cycles
		21	CASC Credit fifo empty
		20	CASC TLB don’t have any Cycles
		19	Miss Fub Done
		18	Read Stream Done
		17	Read Steam Fifo is empty
		16	Recycle Fifo in rstrm is empty
		15	TLB Pend Done
		14	TLB Pend PQ Array Is done
		13	TLB pend PB Array is done
		12	Read route fub is done
		11	Gafm Data fifo is empty
		10	GAP data fifo is empty
		9	GAC data fifo is empty
		8	Wrdp is done with all the cycles
		7	Wrdp RID fifo is empty
		6	No hold from midarb to RTSTRM
		5	No hold from TLBPEND to MIDARB
		4	VTD Mode
		3	Tied to "1" - to be defined
		2	Fence FSM are IDLE
		1	Non PD Load Done
		0	Tied to "1" - to be defined

1.1.19.25 GAC_HWSP_REG - GAC Hardware Status Page Address Register

GAC_HWSP_REG - GAC Hardware Status Page Address Register

Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	04180h-04183h

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. The 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.

DWord	Bit	Description
0	31:12	**GAC HWSP Register** Default Value: 00000h Access: R/W
	11:0	**Reserved Bits** Default Value: 000h Access: RO

1.1.19.26 MEDIA_ENG_FR - Media Engine Fault Register

MEDIA_ENG_FR - Media Engine Fault Register

Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	04194h-04197h

Media Engine Fault Register

DWord	Bit	Description
0	31:12	**Virtual Address of Fault** Default Value: 00000h Access: R/W This is the original Address of the Page that generated the First fault for this engine. This value is locked and not updated on subsequent faults, until the valid bit of this register is cleared by SW

MEDIA_ENG_FR - Media Engine Fault Register

	11	**GTTSEL** Default Value: 0b Access: R/W This bit indicates if the valid bit happened while using PPGTT or GGTT: 0 - PPGTT, 1 - GGTT This value is locked and not updated on subsequent faults, until the valid bit of this register is cleared by SW
	10:3	**SRCID of Fault** Default Value: 00h Access: R/W This is the Source ID of the unit that requested the cycle that generated the First Page fault for this engine. This value is locked and not updated on subsequent faults, until the valid bit of this register is cleared by SW
	2:1	**Fault Type** Default Value: 00b Access: R/W Type of Fault recorded: 00 - Page Fault. 01 - Invalid PD Fault 10 - Unloaded PD Fault 11 - Invalid and Unloaded PD fault This value is locked and not updated on subsequent faults, until the valid bit of this register is cleared by SW
	0	**Valid Bit** Default Value: 0b Access: R/W This bit indicates that the first fault for this engine has been recorded. It can only be cleared by SW, which will also clear the other fields.

1.1.19.27 GAB_HWSP_REG - GAB Hardware Status Page Address Register

GAB_HWSP_REG - GAB Hardware Status Page Address Register	
Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	04280h-04283h

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. The 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.

DWord	Bit	Description
0	31:12	**GAB HWSP Register** Default Value: 00000h Access: R/W
	11:0	**Reserved Bits** Default Value: 000h Access: RO

1.1.19.28 BLT_ENG_FR - Blitter Engine Fault Register

BLT_ENG_FR - Blitter Engine Fault Register	
Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	04294h-04297h

Blitter Engine Fault Register

DWord	Bit	Description
0	31:12	**Virtual Address of Page Fault** Default Value: 00000h Access: R/W This is the original Address of the Page that generated the First fault for this engine. This value is locked and not updated on subsequent faults, until the valid bit of this register is cleared by SW

BLT_ENG_FR - Blitter Engine Fault Register

	11	**Blitter GTTSEL** Default Value: 0b Access: R/W This bit indicates if the valid bit happened while using PPGTT or GGTT: 0 - PPGTT, 1 - GGTT This value is locked and not updated on subsequent faults, until the valid bit of this register is cleared by SW
	10:3	**SRCID of Fault** Default Value: 00h Access: R/W This is the Source ID of the unit that requested the cycle that generated the First Page fault for this engine. This value is locked and not updated on subsequent faults, until the valid bit of this register is cleared by SW
	2:1	**Fault Type** Default Value: 00b Access: R/W Type of Fault recorded: 00 - Page Fault. 01 - Invalid PD Fault 10 - Unloaded PD Fault 11 - Invalid and Unloaded PD fault This value is locked and not updated on subsequent faults, until the valid bit of this register is cleared by SW
	0	**Valid Bit** Default Value: 0b Access: R/W This bit indicates that the first fault for this engine has been recorded. It can only be cleared by SW, which will also clear the other fields.

1.1.19.29 TLB_RD_ADDR - TLB_RD_ADDRESS Register

<table>
<tr><th colspan="3">TLB_RD_ADDR - TLB_RD_ADDRESS Register</th></tr>
<tr><td colspan="3">Register Space: MMIO: 0/2/0
Default Value: 0x00000000</td></tr>
<tr><td colspan="3">Address: 04700h-04703h</td></tr>
<tr><td colspan="3">TLB Read Address</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:10</td><td><b>Reserved Bits</b>
<table>
<tr><td>Default Value:</td><td>0000000000000000000000b</td></tr>
<tr><td>Access:</td><td>RO</td></tr>
</table>
Reserved Bits</td></tr>
<tr><td></td><td>9:0</td><td><b>TLB Read Address</b>
<table>
<tr><td>Default Value:</td><td>0000000000b</td></tr>
<tr><td>Access:</td><td>R/W</td></tr>
</table>
TLB Read Address
MSB<9:X> :
<table>
<tr><td>TLB Select</td><td><9:X></td><td>PAT MSB: Section of the PAT used.</td></tr>
<tr><td>PAT_MSB_VLFTLB</td><td>00000</td><td>32 entries - 32</td></tr>
<tr><td>PAT_MSB_CVSTLB</td><td>00001</td><td>32 entries - 32</td></tr>
<tr><td>PAT_MSB_RCCTLB</td><td>0001</td><td>64 entries - 64</td></tr>
<tr><td>PAT_MSB_ZTLB</td><td>001</td><td>128 entries - 128</td></tr>
<tr><td>PAT_MSB_L3TLB</td><td>01</td><td>160 entries - 256</td></tr>
<tr><td>PAT_MSB_CASCTLB</td><td>10</td><td>140 entries - 256</td></tr>
</table>
LSB <X:0> :
GEN RAM ADDRES in Selected TLB</td></tr>
</table>

1.1.19.30 TLB_RD_DATA - TLB_RD_DATA Register

TLB_RD_DATA - TLB_RD_DATA Register	
Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	04704h-04707h

TLB_READ_DATA Register

DWord	Bit	Description
0	31:0	**TLB_READ_DATA Register** Default Value: 00000000h Access: RO Return data

1.1.19.31 VLFTLB_VLD_0 - Valid Bit Vector 0 for VLF

VLFTLB_VLD_0 - Valid Bit Vector 0 for VLF	
Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	04720h-04723h

This register contains the valid bits for entries 0-31 of VLFTLB

DWord	Bit	Description
0	31:0	**Valid Bit Vector 0 for VLF** Default Value: 00000000h Access: RO Valid bits per entry

1.1.19.32 CVSTLB_VLD_0 - Valid Bit Vector 0 for CVS

CVSTLB_VLD_0 - Valid Bit Vector 0 for CVS	
Register Space:	MMIO: 0/2/0
Default Value:	0x00000000

CVSTLB_VLD_0 - Valid Bit Vector 0 for CVS	
Address:	04724h-04727h

This register contains the valid bits for entries 0-31 of CVSTLB

DWord	Bit	Description
0	31:0	**Valid Bit Vector 0 for CVS** Default Value: 00000000h Access: RO Valid bits per entry

1.1.19.33 RCCTLB_VLD_0 - Valid Bit Vector 0 for RCC

RCCTLB_VLD_0 - Valid Bit Vector 0 for RCC	
Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	04728h-0472Bh

This register contains the valid bits for entries 0-31 of RCCTLB

DWord	Bit	Description
0	31:0	**Valid Bit Vector 0 for RCC** Default Value: 00000000h Access: RO Valid bits per entry

1.1.19.34 RCCTLB_VLD_1 - Valid Bit Vector 1 for RCC

RCCTLB_VLD_1 - Valid Bit Vector 1 for RCC	
Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	0472Ch-0472Fh

This register contains the valid bits for entries 0-31 of RCCTLB

DWord	Bit	Description
0	31:0	**Valid Bit Vector 1 for RCC** Default Value: 00000000h Access: RO Valid bits per entry

1.1.19.35 ZTLB_VLD_0 - Valid Bit Vector 0 for Z

ZTLB_VLD_0 - Valid Bit Vector 0 for Z

Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	04730h-04733h

This register contains the valid bits for entries 0-31 of ZTLB

DWord	Bit	Description
0	31:0	**Valid Bit Vector 0 for Z** Default Value: 00000000h Access: RO Valid bits per entry

1.1.19.36 ZTLB_VLD_1 - Valid Bit Vector 1 for Z

ZTLB_VLD_1 - Valid Bit Vector 1 for Z

Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	04734h-04737h

This register contains the valid bits for entries 0-31 of ZTLB

DWord	Bit	Description
0	31:0	**Valid Bit Vector 1 for Z** Default Value: 00000000h Access: RO Valid bits per entry

1.1.19.37 ZTLB_VLD_2 - Valid Bit Vector 2 for Z

ZTLB_VLD_2 - Valid Bit Vector 2 for Z	
Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	04738h-0473Bh

This register contains the valid bits for entries 0-31 of ZTLB

DWord	Bit	Description
0	31:0	**Valid Bit Vector 2 for Z** Default Value: 00000000h Access: RO Valid bits per entry

1.1.19.38 ZTLB_VLD_3 - Valid Bit Vector 3 for Z

ZTLB_VLD_3 - Valid Bit Vector 3 for Z	
Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	0473Ch-0473Fh

This register contains the valid bits for entries 0-31 of ZTLB

DWord	Bit	Description
0	31:0	**Valid Bit Vector 3 for Z** Default Value: 00000000h Access: RO Valid bits per entry

1.1.19.39 L3TLB_VLD_0 - Valid Bit Vector 0 for L3

L3TLB_VLD_0 - Valid Bit Vector 0 for L3		
Register Space:		MMIO: 0/2/0
Default Value:		0x00000000
Address:	04740h-04743h	
This register contains the valid bits for entries 0-31 of L3TLB		
DWord	**Bit**	**Description**
0	31:0	**Valid Bit Vector 0 for L3** Default Value: 00000000h Access: RO Valid bits per entry

1.1.19.40 L3TLB_VLD_1 - Valid Bit Vector 1 for L3

L3TLB_VLD_1 - Valid Bit Vector 1 for L3		
Register Space:		MMIO: 0/2/0
Default Value:		0x00000000
Address:	04744h-04747h	
This register contains the valid bits for entries 0-31 of L3TLB		
DWord	**Bit**	**Description**
0	31:0	**Valid Bit Vector 1 for L3** Default Value: 00000000h Access: RO Valid bits per entry

1.1.19.41 L3TLB_VLD_2 - Valid Bit Vector 2 for L3

L3TLB_VLD_2 - Valid Bit Vector 2 for L3	
Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	04748h-0474Bh

This register contains the valid bits for entries 0-31 of L3TLB

DWord	Bit	Description
0	31:0	**Valid Bit Vector 2 for L3** Default Value: 00000000h Access: RO Valid bits per entry

1.1.19.42 L3TLB_VLD_3 - Valid Bit Vector 3 for L3

L3TLB_VLD_3 - Valid Bit Vector 3 for L3	
Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	0474Ch-0474Fh

This register contains the valid bits for entries 0-31 of L3TLB

DWord	Bit	Description
0	31:0	**Valid Bit Vector 3 for L3** Default Value: 00000000h Access: RO Valid bits per entry

1.1.19.43 L3TLB_VLD_4 - Valid Bit Vector 4 for L3

L3TLB_VLD_4 - Valid Bit Vector 4 for L3	
Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	04750h-04753h

This register contains the valid bits for entries 0-31 of L3TLB

DWord	Bit	Description
0	31:0	**Valid Bit Vector 4 for L3** Default Value: 00000000h Access: RO Valid bits per entry

1.1.19.44 L3TLB_VLD_5 - Valid Bit Vector 5 for L3

L3TLB_VLD_5 - Valid Bit Vector 5 for L3	
Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	04754h-04757h

This register contains the valid bits for entries 0-31 of L3TLB

DWord	Bit	Description
0	31:0	**Valid Bit Vector 5 for L3** Default Value: 00000000h Access: RO Valid bits per entry

1.1.19.45 L3TLB_VLD_6 - Valid Bit Vector 6 for L3

L3TLB_VLD_6 - Valid Bit Vector 6 for L3

Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	04758h-0475Bh

This register contains the valid bits for entries 0-31 of L3TLB

DWord	Bit	Description
0	31:0	**Valid Bit Vector 6 for L3** Default Value: 00000000h Access: RO Valid bits per entry

1.1.19.46 L3TLB_VLD_7 - Valid Bit Vector 7 for L3

L3TLB_VLD_7 - Valid Bit Vector 7 for L3

Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	0475Ch-0475Fh

This register contains the valid bits for entries 0-31 of L3TLB

DWord	Bit	Description
0	31:0	**Valid Bit Vector 7 for L3** Default Value: 00000000h Access: RO Valid bits per entry

1.1.19.47 CASCTLB_VLD_0 - Valid Bit Vector 0 for CASC

CASCTLB_VLD_0 - Valid Bit Vector 0 for CASC	
Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	04760h-04763h

This register contains the valid bits for entries 0-31 of CASCTLB

DWord	Bit	Description
0	31:0	**Valid Bit Vector 0 for CASC** Default Value: 00000000h Access: RO Valid bits per entry

1.1.19.48 CASCTLB_VLD_1 - Valid Bit Vector 1 for CASC

CASCTLB_VLD_1 - Valid Bit Vector 1 for CASC	
Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	04764h-04767h

This register contains the valid bits for entries 0-31 of CASCTLB

DWord	Bit	Description
0	31:0	**Valid Bit Vector 1 for CASC** Default Value: 00000000h Access: RO Valid bits per entry

1.1.19.49 CASCTLB_VLD_2 - Valid Bit Vector 2 for CASC

CASCTLB_VLD_2 - Valid Bit Vector 2 for CASC	
Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	04768h-0476Bh

This register contains the valid bits for entries 0-31 of CASCTLB

DWord	Bit	Description
0	31:0	**Valid Bit Vector 2 for CASC** Default Value: 00000000h Access: RO Valid bits per entry

1.1.19.50 CASCTLB_VLD_3 - Valid Bit Vector 3 for CASC

L3TLB_VLD_3 - Valid Bit Vector 3 for L3	
Register Space:	MMIO: 0/2/0
Default Value:	0x00000000
Address:	0474Ch-0474Fh

This register contains the valid bits for entries 0-31 of L3TLB

DWord	Bit	Description
0	31:0	**Valid Bit Vector 3 for L3** Default Value: 00000000h Access: RO Valid bits per entry

1.1.19.51 CASCTLB_VLD_4 - Valid Bit Vector 4 for CASC

CASCTLB_VLD_4 - Valid Bit Vector 4 for CASC		
Register Space:		MMIO: 0/2/0
Default Value:		0x00000000
Address:	04770h-04773h	

This register contains the valid bits for entries 0-31 of CASCTLB

DWord	Bit	Description
0	31:0	**Valid Bit Vector 4 for CASC** Default Value: 00000000h Access: RO Valid bits per entry

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 Ivy Bridge 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_SEMAPHORE_MBOX	Memory write
MI_STORE_DATA_IMM	Always
MI_STORE_DATA_INDEX	Always
MI_LOAD_REGISTER_IMM	Always
MI_UPDATE_GTT	Always

Command	Qualifications
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

Source:	RenderCS
Length Bias:	1

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
- This instruction can be in either a ring buffer or batch buffer.
- For pre-emption, the wrap count in the ring buffer head register is no longer maintained by hardware. The hardware updates the wrap count to the value in the UHPTR register.

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

1.2.4 MI_ARB_ON_OFF

MI_ARB_ON_OFF

Source:	RenderCS
Length Bias:	1

The MI_ARB_ON_OFF instruction is used to disable/enable context switching. Note that context switching will remain disabled until re-enabled through use of this command.

This command will also prevent a switch in the case of waiting on events, running out of commands. 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-privileged batch buffer.

DWord	Bit	Description

MI_ARB_ON_OFF

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** 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

Source:	RenderCS
Length Bias:	1

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

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

1.2.6 MI_CONDITIONAL_BATCH_BUFFER_END

MI_CONDITIONAL_BATCH_BUFFER_END

Source:	RenderCS
Length Bias:	2

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

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

MI_CONDITIONAL_BATCH_BUFFER_END

		Format: OpCode
	28:23	**MI Command Opcode** Default Value: 36h MI_CONDITIONAL_BATCH_BUFFER_END Format: OpCode
	22	**Use Global GTT** Default Value: 0h Format: U1 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**.
	21	**Compare Semaphore** Default Value: 0h Format: U1 If set, the value from the Compare Data Dword is compared to the value from the Compare Address in memory. If the value at Compare Address is greater than the Compare Data Dword, execution of current command buffer should continue. If clear, no comparison takes place.
	19:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Format: =n Total Length - 2. Excludes DWord (0,1).
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** Format: MBZ

1.2.7 MI_BATCH_BUFFER_START (Render)

MI_BATCH_BUFFER_START

Source:	RenderCS
Length Bias:	2

The MI_BATCH_BUFFER_START command is used to initiate the execution of commands stored in a batch buffer.

MI_BATCH_BUFFER_START

For restrictions on the location of batch buffers, see Batch Buffers in the Device Programming Interface chapter of MI Functions.

Programming Notes

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
	21:17	**Reserved**	
		Format:	MBZ
	16	**Reserved**	
		Format:	MBZ
	15	**Reserved**	
		Format:	MBZ
	14	**Reserved**	
		Format:	MBZ
	13	**Reserved**	
		Format:	MBZ
	11	**Clear Command Buffer Enable**	
		Format:	Enable
		The address of the batch buffer is an offset into the WOPCM area. This batch buffer needs to be preceded by a MI_FLUSH command or PIPE_CONTROL with CS Stall set.	
	10	**Reserved**	
		Format:	MBZ
	8	**Address Space Indicator**	
		Description	**Project**
		SW must ensure the "Address Space Indicator" of the chained batch buffer to be same as the initial batch buffer. Ex: If the MI_BATCH_BUFFER_START executed from Ring Buffer has "Address Space Indicator" as "PPGTT" then all subsequent chained batch buffers (not second level Batch Buffers) must be in "PPGTT". Not complying to above programming will result in unknown behavior of HW. Second level batch buffer can select its "Address space Indicator" independent of the parent batch buffer.	
		This field must be '0' unless the Per-Process GTT Enable is '1'	

<table>
<tr><th colspan="3">MI_BATCH_BUFFER_START</th></tr>
<tr><td></td><td></td><td><table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>GGTT</td><td>This batch buffer will be accessed via the GGTT.</td></tr><tr><td>1h</td><td>PPGTT</td><td>This batch buffer will be accessed via the PPGTT.</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>0h Excludes DWord (0,1)</td></tr><tr><td>Format:</td><td>=n Total - Bias</td></tr></table></td></tr>
<tr><td>1</td><td>31:2</td><td><b>Batch Buffer Start Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:2]BatchBuffer</td></tr></table>This field specifies Bits 31:2 of the starting address of the batch buffer.</td></tr>
<tr><td></td><td>1:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
</table>

1.2.7.1 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.

“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.

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.

2.1.7.2 Privileged Commands

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

Privileged Commands

Privileged Command	Function in non-privileged batch buffers
MI_LOAD_REGISTER_IMM	Byte enables are turned off
MI_UPDATE_GTT	Byte enabled are turned off
MI_STORE_DATA_IMM	Command is translated using the Per-process GTT if **Per-Process Virtual Address Space is set**
MI_STORE_DATA_INDEX	
MI_STORE_REGISTER_MEM	Command is translated and completed with byte enables turned off
MI_DISPLAY_FLIP	Command is ignored by the hardware

Parsing one of the commands in the table above from a non-secure batch buffer will flag an error and convert the command to a NOOP.

1.2.7.2 User Mode Privileged Commands

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

User Mode Privileged Commands

User Mode Privileged Command	Function in non-privileged batch buffers
MI_LOAD_REGISTER_IMM	Command is converted to NOOP
MI_UPDATE_GTT	Command is converted to NOOP
MI_STORE_DATA_IMM	Command is converted to NOOP if **Use Global GTT** is enabled.
MI_STORE_DATA_INDEX	Command is converted to NOOP if **Use Global GTT** is enabled.
MI_STORE_REGISTER_MEM	Command is converted to NOOP
MI_DISPLAY_FLIP	Command is converted to NOOP
MI_ARB_ON_OFF	Command is converted to NOOP
MI_ARB_CHECK	Command is converted to NOOP
MI_WAIT_FOR_EVENT	Command is converted to NOOP

1.2.8 MI_CLFLUSH

MI_CLFLUSH

Source:	RenderCS
Length Bias:	2

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.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>0h MI_COMMAND</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:23</td><td><b>MI Command Opcode</b><table><tr><td>Default Value:</td><td>27h Store DW MI_CLFLUSH</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>22</td><td><b>Use Global GTT</b>
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 <b>Per Process GTT Enable</b> bit is clear.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Per Process Graphics Address</td><td></td></tr><tr><td>1h</td><td>Global Graphics Address</td><td>This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.</td></tr></table></td></tr>
<tr><td></td><td>21:10</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>9:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>0h</td></tr><tr><td>Format:</td><td>=n Total Length - 2. Excludes DWord (0,1).</td></tr></table></td></tr>
<tr><td>1</td><td>31:12</td><td><b>Page Base Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:12]</td></tr></table>4KB aligned Page Address which software requires hardware to flush to DRAM.</td></tr>
<tr><td></td><td>11:6</td><td><b>Starting Cacheline Offset</b><table><tr><td>Format:</td><td>U6 Zero based starting cacheline offset to the Page Base Address.</td></tr></table></td></tr>
<tr><td></td><td>5:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>2</td><td>31:16</td><td><b>Address</b></td></tr>
<tr><td></td><td>15:0</td><td><b>Page Base Address</b><table><tr><td>Format:</td><td>GraphicsAddress[47:32]</td></tr></table>This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host’s 64-bit virtual address space.</td></tr>
<tr><td>3..n</td><td>31:0</td><td><b>DW Representing ½ Cache Line</b></td></tr>
</table>

MI_CLFLUSH

Format:	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.

Programming Notes

Always even number of "DW Representing 1/2 cacheline" terms must be programmed.

1.2.9 MI_DISPLAY_FLIP

MI_DISPLAY_FLIP

Source:	RenderCS
Length Bias:	2

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

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 PIPE_CONTROL command does not guarantee that outstanding flip operations have completed). The MI_WAIT_FOR_EVENT command must be used to provide this synchronization to avoid back to back MI_DISPLAY_FLIP commands to the same display plane – 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.

After a display flip operation is requested, software is responsible for initiating any required synchronization with subsequent buffer clear or rendering operations. For multi-buffering (e.g., double buffering) operations, this will typically require updating SURFACE_STATE or the binding table to change the rendering (back) buffer. In addition, prior to any subsequent clear or rendering operations, software must typically ensure that the new rendering buffer is not actively being displayed. Again, the MI_WAIT_FOR_EVENT command or Display Flip Pending hardware status can be used to provide this synchronization. See Display Flip Synchronization in the Device Programming Interface chapter of MI Functions.

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 The display buffer command uses the linear dword offset for the linear buffers from the Display Interface registers. Software is allowed to change the offset via the MMIO interface irrespective of the flip commands enqueued in the command stream. For linear buffers, the display subsystem uses the dword offset in generation of the final request to memory. For synchronous flips the offset is updated on the next vblank. It is not necessary to have a sync flip enqueued to update the dword offset. Linear memory does not support asynchronous flips DWord 3 (Left Eye Display Buffer Base Address) must not be set with synchronous flips or asynchronous 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**		
		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.		
	21:19	**Display (Plane) Select**		
		Format:	U3	
		This field selects which display plane is to perform the flip operation.		
		Value	**Name**	**Project**
		0h	Display Plane A	
		1h	Display Plane B	
		2h	Display Sprite A	
		3h	Display Sprite B	
		4h	Display Plane C	
		5h	Display Sprite C	
	18:8	**Reserved**		
		Format:	MBZ	
	7:0	**DWord Length**		
		Format:	=n	
		Total Length - 2. Excludes DWord (0,1).		
		For Synchronous Flips and Asynchronous Flips, this field must be programmed to 1h for a total length of 3.		
		Value	**Name**	
		0h	**[Default]**	
		1h	For Synchronous Flips and Asynchronous Flips	
1	31	**Reserved**		
		Format:	MBZ	
	30:16	**Reserved**		
		Format:	MBZ	
	15:6	**Display Buffer Pitch**		
		Default Value:	0h	
		Format:	U10	
		For Synchronous Flips, this field specifies the 64-byte aligned pitch of the new display buffer. For Asynchronous Flips, this parameter is programmed so that all the flips in a flip chain should maintain the same pitch as programmed with the last synchronous flip or direct through MMIO.		
	5:1	**Reserved**		

MI_DISPLAY_FLIP

Format:	MBZ

0 **Tile Parameter**

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
0h	Linear **[Default]**	For Syncronous Flips Only
1h	Tiled X	

2 31:12 **Display Buffer Base Address**

Format:	GraphicsAddress[31:12]

This field specifies Bits 31:12 of the Graphics Address of the new display buffer.

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

11:3 **Reserved**

Format:	MBZ

1:0 **Flip Type**

This field specifies whether the flip operation should be performed asynchronously to vertical retrace.

Value	Name	Description
00b	Sync Flip **[Default]**	The flip will occur during the vertical blanking interval – thus avoiding any tearing artifacts.
01b	Async Flip	The flip will occur "as soon as possible" – and may exhibit tearing artifacts
11b	Reserved	

Programming Notes
Asynch flips are Supported on X-Tiled Frame buffers only.
For Asynch Flips the Buffers used must be 32KB aligned.

1.2.10 MI_FLUSH

MI_FLUSH

Source:	RenderCS
Length Bias:	1

Description	Project
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:	

MI_FLUSH

Flush any dirty data in the Render Cache to memory. This is done by default, however this can be inhibited.

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.

In order to use this command, bit 12 in the MI_MODE(0x209c) must be enabled.

If GFX_MODE(0x229C) bit 13, this command will cause a config write to MMIO register space with the address 0x4f100.

MI_FLUSH command is no longer validated or supported. Use at your own risk.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>0h MI_COMMAND</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:23</td><td><b>MI Command Opcode</b><table><tr><td>Default Value:</td><td>04h MI_FLUSH</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>22:7</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>5</td><td><b>Indirect State Pointers Disable</b><table><tr><td>Format:</td><td>Disable</td></tr></table>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.</td></tr>
<tr><td></td><td>4</td><td><b>Generic Media State Clear</b><table><tr><td></td><td></td></tr><tr><td>Format:</td><td>Disable</td></tr></table>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.</td></tr>
<tr><td></td><td>3</td><td><b>Global Snapshot Count Reset</b><table><tr><td>Format:</td><td>Boolean</td></tr></table>The Statistics Counters are also reset; SW should never set this bit during normal operation since the Statistics Counters are intended to be free running.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Don’t Reset</td><td>Do not reset the snapshot counts or Statistics Counters.</td></tr><tr><td>1h</td><td>Reset</td><td>Reset the snapshot count for all the units and reset the Statistics Counters except as noted above.</td></tr></table></td></tr>
</table>

MI_FLUSH

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

2 **Render Cache Flush Inhibit**

Format:	Boolean

If set, the Render Cache is not flushed as part of the processing of this command.

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

1 **State/Instruction Cache Invalidate**

Format:	Boolean

If set, Invalidates the State and Instruction Cache

Value	Name	Description
0h	Don't Invalidate	Leave State/Instruction Cache unaffected
1h	Invalidate	Invalidate State/Instruction Cache

0 **Reserved**

Format:	MBZ

1.2.11 MI_LOAD_REGISTER_IMM

MI_LOAD_REGISTER_IMM

Source:	RenderCS
Length Bias:	2

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	Project
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	

MI_LOAD_REGISTER_IMM

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.

MI_LOAD_REGISTER_IMM command to program Scanline Register followed by Wait For Event command with Scanline Wait, should always be programmed in the same cacheline together without any commands (including pipe control) in between and also should be submitted in the same ring dispatch.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 22h MI_LOAD_REGISTER_IMM Format: OpCode
	22:12	**Reserved** 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 '1111b', then this command will behave as a NOOP. Otherwise, the value is forwarded to the destination register.
	7:0	**DWord Length** Default Value: 1h Format: =n Total Length - 2. Excludes DWord (0,1).
1	31:2	**Register Offset** Format: 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 the 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** 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	
Source:	RenderCS
Length Bias:	1

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

Programming Notes	Project
Performance : The MI_NOOP process time is reduced to 1 clock. An example use of the improved NOOP throughput is for some multi-pass media applications where some unwanted media object commands are replaced by MI_NOOP commands without repacking the commands in a batch buffer.	

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b>
Default Value: 0h MI_COMMAND</td></tr>
<tr><td></td><td>28:23</td><td><b>MI Command Opcode</b>
Default Value: 0h MI_NOOP</td></tr>
<tr><td></td><td>22</td><td><b>Identification Number Register Write Enable</b>
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.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Disable</td><td>Do not write the NOP_ID register.</td></tr>
<tr><td>1h</td><td>Enable</td><td>Write the NOP_ID register.</td></tr>
</table></td></tr>
<tr><td></td><td>21:0</td><td><b>Identification Number</b>
Format: U22
This field contains a 22-bit number which can be written to the MI NOPID register.</td></tr>
</table>

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	
Source:	RenderCS
Length Bias:	1

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.

MI_REPORT_HEAD

Programming Notes

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

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

1.2.15 MI_SEMAPHORE_MBOX

MI_SEMAPHORE_MBOX

Source:	RenderCS
Length Bias:	2

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 two 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_sli 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** 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	

MI_SEMAPHORE_MBOX

<table>
<tr><td></td><td></td><td>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.
<table><tr><th>Programming Notes</th></tr><tr><td>This field is only valid when Compare Register Field is reset.</td></tr></table></td></tr>
<tr><td></td><td>21</td><td><b>Update Semaphore</b>
If set, the value from the <b>Semaphore Data Dword</b> is written to memory. If <b>Compare Semaphore</b> is also set, the semaphore is not updated if the semaphore comparison fails. If clear, the data at <b>Semaphore Address</b> is not changed.
<table><tr><th>Programming Notes</th></tr><tr><td>This field should be always clear when Compare Register Field is set.</td></tr></table></td></tr>
<tr><td></td><td>20</td><td><b>Compare Semaphore</b>
If set, the value from the <b>Semaphore Data Dword</b> is compared to the value from the <b>Semaphore Address</b> in memory when Compare Register is clear. If set, the value from the <b>Semaphore Data Dword</b> is compared to the value from <b>MMIO Register</b> selected by <b>Register Select</b> field when Compare Register is set. If the value at <b>Semaphore Address/MMIO Register is greater than the Semaphore Data Dword</b>, execution is continued from the current command buffer. If clear, no comparison takes place. <b>Update Semaphore</b><i>must</i> be set in this case.</td></tr>
<tr><td></td><td>19</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>18</td><td><b>Compare Register</b>
If set, data in MMIO register will be used for compare. If clear, data in memory will be used for compare.
<table><tr><th>Programming Notes</th></tr><tr><td>Compare Register field should be always set.</td></tr></table></td></tr>
<tr><td></td><td>17:16</td><td><b>Register Select</b>
<table><tr><td>Format:</td><td>Register Select</td></tr></table>
If <b>Compare Register</b> is set in bit[18], this field indicates which register will be used.
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>RVSYNC</td><td>VCS Register</td></tr>
<tr><td>2h</td><td>RBSYNC</td><td>BCS Register</td></tr>
<tr><td>3h</td><td>Use General Register Select</td><td></td></tr></table></td></tr>
<tr><td></td><td>15:14</td><td><b>Reserved</b>
<table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>13:8</td><td><b>Reserved</b>
<table><tr><td></td><td></td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b>
<table><tr><td>Default Value:</td><td>0h</td></tr><tr><td>Format:</td><td>=n Total Length - 2. Excludes DWord (0,1).</td></tr></table></td></tr>
<tr><td>1</td><td>31:0</td><td><b>Semaphore Data Dword</b>
<table><tr><td>Format:</td><td>U32</td></tr></table>
Data dword to compare/update memory. The Data dword is supplied by software to control execution of the command buffer. If the compare is enabled and the data at Semaphore Address is greater than this dword, the execution of the command buffer continues.</td></tr>
<tr><td>2</td><td>31:2</td><td><b>PointerBitFieldName/MMIO Register Address</b>
<table><tr><td>Format:</td><td>GraphicsVirtualAddress[31:2]Semaphore</td></tr></table></td></tr>
</table>

MI_SEMAPHORE_MBOX		
		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** Format: MBZ

1.2.16 MI_SET_CONTEXT

MI_SET_CONTEXT

Source:	RenderCS
Length Bias:	2

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 saves the current HW context values to the current logical context address, and then restores (loads) 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 NOOP. **Specific to the Render command stream only.**

This command also includes some controls over the context save/restore process.

- The **Force Restore** bit can be used to refresh the on-chip device state from the same memory address if the indirect state buffers have been modified.
- The **Restore Inhibit** bit can be used to prevent the new context from being loaded at all. This must be used to prevent an uninitialized context from being loaded. Once software has initialized a context (by setting all state variables to initial values via commands), the context can then be stored and restored normally.
- This command needs to be always followed by a single MI_NOOP instruction to workaround a 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.
- MI_SET_CONTEXT commands are permitted only within a ring buffer (not within a batch buffer).

Programming Notes	Project
Workaround : If **Flush TLB Invalidation Mode** is enabled it is 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.	
MI_ARB_ON_OFF with 'Arbitration Enable Reset' set should be programmed before an MI_SET_CONTEXT command. MI_ARB_ON_OFF with 'Arbitration Enable' set should be programmed after an MI_SET_CONTEXT command. This programming ensures that PSMI context switch flows do not conflict with MI_SET_CONTEXT flows.	

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** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Format: =n Total Length - 2. Excludes DWord (0,1).
1	31:12	**Logical Context Address**

		Format: GraphicsAddress[31:12]LogicalContext Description | Project This field contains the 4KB-aligned graphics memory 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. This field needs to be 4KB aligned virtual address.
	11:10	**Reserved** Format: MBZ
	9	**Reserved** Format: MBZ
	8	**Reserved, Must be 1** Format: Must Be One
	7:5	**Reserved** Format: MBZ
	4	**Reserved** Format: MBZ
	3	**Extended State Save Enable** 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. 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)
	2	**Extended State Restore Enable** Format: Enable 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)
	1	**Force Restore** 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

MI_SET_CONTEXT		
		processing of this command.
	0	**Restore Inhibit** 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
Source:	RenderCS
Length Bias:	2

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-privilege" batch buffer to access global virtual space, doing so will be treated as privilege access violation. Refer “User Mode Privilege Command” in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation. This command can be used within ring buffers and/or privilege batch buffers to access global virtual space.

This command can be used for general software synchronization through variables in cacheable memory (i.e., where software does not need to poll un-cached memory or device registers).

This command simply initiates the write operation with command execution proceeding normally. Although the write operation is guaranteed to complete eventually, there is no mechanism to synchronize command execution with the completion (or even initiation) of these operations.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 20h MI_STORE_DATA_IMM Format: OpCode
	22	**Use Global GTT** Project: All Format: Boolean If set, this command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer. If clear, the PPGTT will be used. It is allowed for this bit to be clear when executing this command from a privileged (secure) batch buffer. This bit must be ‘1’ if the Per Process GTT Enable bit is clear.

MI_STORE_DATA_IMM

	21	**Reserved** Format: MBZ
	20:10	**Reserved** Format: MBZ
	9:0	**DWord Length** Default Value: 2h Format: =n Total Length - 2. Excludes DWord (0,1) **Programming Notes** DWord Length programmed must not exceed 0x3FE
1	31:0	**Reserved** Format: MBZ
2	31:2	**Address** Format: GraphicsAddress[31:2]U32(2) This field specifies Bits 31:2 of the Address where the DWord will be stored. As the store address must be DWord-aligned, Bits 1:0 of that address MBZ. This address must be 8B aligned for a store "QW" command.
	1:0	**Reserved** Format: MBZ
3	31:0	**Data DWord 0** 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** 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

Source:	RenderCS
Length Bias:	2

The MI_STORE_DATA_INDEX command requests a write of the data constant supplied in the packet to the specified offset from the System Address defined by the Hardware Status Page Address Register. As the write

MI_STORE_DATA_INDEX

targets a System Address, the write operation is coherent with the CPU cache (i.e., the processor cache is snooped).

Programming Notes

-

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

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 21h MI_STORE_DATA_INDEX Format: OpCode
	22	**Reserved**
	21	**Reserved** Format: MBZ
	20:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 1h Format: =n Total Length - 2. Excludes DWord (0,1) = 1 for DWord, 2 for QWord.
1	31:12	**Reserved** Format: MBZ
	11:2	**Offset** Format: U10 zero-based DWord offset into the HW status page. Format: HardwareStatusPageOffset[11:2]U32 This field specifies the offset (into the hardware status page) to which the data will be written. Note that the first few DWords of this status page are reserved for special-purpose data storage – targeting these reserved locations via this command is UNDEFINED. This address must be 8B aligned for a store QW command. Value: [16, 1023] Name:
	1:0	**Reserved** Format: MBZ
2	31:0	**Data DWord 0** Format: U32 This field specifies the DWord value to be written to the targeted location.For a QWord write this

MI_STORE_DATA_INDEX		
		DWord is the lower DWord of the QWord to be reported (DW 0).
3	31:0	**Data DWord 1** 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

<table>
<tr><th colspan="3">MI_STORE_REGISTER_MEM</th></tr>
<tr><td colspan="2">Project:</td><td>All</td></tr>
<tr><td colspan="2">Source:</td><td>RenderCS</td></tr>
<tr><td colspan="2">Length Bias:</td><td>2</td></tr>
<tr><td colspan="3">The MI_STORE_REGISTER_MEM command requests a register read from a specified memory mapped register location in the device and store of that DWord to memory. The register address is specified along with the command to perform the read.</td></tr>
<tr><th colspan="3">Programming Notes</th></tr>
<tr><td colspan="3">The command temporarily halts command execution.
The memory address for the write is snooped on the host bus.
This command should not be used from within a "non-privilege" batch buffer to access global virtual space. doing so will be treated as privilege access violation. Refer "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation. This command can be used within ring buffers and/or "privilege" batch buffers to access global virtual space.
This command will cause undefined data to be written to memory if given register addresses for the PGTBL_CTL_0 or FENCE registers.</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>0h MI_COMMAND</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:23</td><td><b>MI Command Opcode</b><table><tr><td>Default Value:</td><td>24h MI_STORE_REGISTER_MEM</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>22</td><td><b>Use Global GTT</b>
It is allowed for this bit to be set when executing this command from a privileged (secure) batch or ring buffer. This bit must be clear when programmed from within a non-privileged 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>Per Process Graphics Address</td><td></td><td></td></tr><tr><td>1h</td><td>Global Graphics</td><td>This command will use the global GTT to translate the Address and</td><td></td></tr></table></td></tr>
</table>

MI_STORE_REGISTER_MEM

		Address | this command must be executing from a privileged (secure) batch buffer.
	21	**Reserved** Format: MBZ
	20:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 1h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:26	**Reserved** Format: MBZ
	25:2	**Register Address** Format: MMIOAddress[25:2]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** • Storing a VGA register is not permitted and will store an UNDEFINED value. • The values of PGTBL_CTL0 or any of the FENCE registers cannot be stored to memory; UNDEFINED values will be written to memory if the addresses of these registers are specified.
	1:0	**Reserved** Format: MBZ
2	31:2	**Memory Address** Format: GraphicsAddress[31:2]MMIO_Register This field specifies the address of the memory location where the register value specified in the DWord above will be written. The address specifies the DWord location of the data.Range = GraphicsVirtualAddress[31:2] for a DWord register
	1:0	**Reserved** Format: MBZ

1.2.20 MI_SUSPEND_FLUSH

MI_SUSPEND_FLUSH

Source:	RenderCS
Length Bias:	1

Description	Project
Blocks MMIO sync flush or any flushes related to VT-d while enabled.	

Programming Notes	Project
SW must ensure MI_SUSPEND_FLUSH with "Suspend Flush" enabled have a corresponding MI_SUSPEND_FLUSH with "Suspend Flush" disabled in the same ring dispatch. SW must also ensure not	

MI_SUSPEND_FLUSH

to program MI_WAIT_FOR_EVENT command when "Suspend Flush" is 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** Format: MBZ
	0	**Suspend Flush** Format: Enable Description / Project: This field suspends flush due and IOTLB invalidation. Programming Notes / Project: Workaround: Make sure that Suspend Flush Enabled and Suspend Flush Disabled are in the same tail update.

1.2.21 MI_UPDATE_GTT

MI_UPDATE_GTT

Source:	RenderCS
Length Bias:	2

The MI_UPDATE_GTT command is used to update GTT page table entries in a coherent manner and at a predictable place in the command flow.

An MI_FLUSH should be placed before this command, 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 also invalidates 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.

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 any value to MMIO address 0x101008 to ensure system agent TLBs are invalidated before the new pages can be used.

PPGTT updates cannot be done via **MI_UPDATE_GTT**; gfx driver will have to use MI_STORE_DATA_IMM for PPGTT inline updates.

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

<table>
<tr><th colspan="3">MI_UPDATE_GTT</th></tr>
<tr><td></td><td></td><td><table><tr><td>Default Value:</td><td>0h MI_COMMAND</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:23</td><td><b>MI Command Opcode</b><table><tr><td>Default Value:</td><td>23h MI_UPDATE_GTT</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>22</td><td><b>Use Global GTT</b>
Reserved: Must be 1h. Updating Per Process Graphics Address is not supported.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Per Process Graphics Address</td><td></td></tr><tr><td>1h</td><td>Global Graphics Address</td><td></td></tr></table></td></tr>
<tr><td></td><td>21:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b><table><tr><td>Default Value:</td><td>0h</td></tr><tr><td>Format:</td><td>=n Total Length - 2. Excludes DWord (0,1).</td></tr></table></td></tr>
<tr><td>1</td><td>31:12</td><td><b>Entry Address</b><table><tr><td>Format:</td><td>GraphicsAddress[31:12]</td></tr></table>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.</td></tr>
<tr><td></td><td>11:0</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>2..n</td><td>31:0</td><td><b>Entry Data</b><table><tr><td>Format:</td><td>Table Entry</td></tr></table>This Dword becomes the new page table entry. See PPGTT/Global GTT Table Entries (PTEs) in <i>Memory Interface Registers</i>.</td></tr>
</table>

1.2.22 MI_USER_INTERRUPT

<table>
<tr><th colspan="3">MI_USER_INTERRUPT</th></tr>
<tr><td colspan="3"><table><tr><td>Source:</td><td>RenderCS</td></tr><tr><td>Length Bias:</td><td>1</td></tr></table></td></tr>
<tr><td colspan="3">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.</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>0h MI_COMMAND</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:23</td><td><b>MI Command Opcode</b><table><tr><td>Default Value:</td><td>02h MI_USER_INTERRUPT</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
</table>

MI_USER_INTERRUPT		
	22:0	**Reserved** Format: MBZ

1.2.23 MI_WAIT_FOR_EVENT

MI_WAIT_FOR_EVENT	
Source:	RenderCS
Length Bias:	1

Description	Project
The MI_WAIT_FOR_EVENT command is used to pause command stream processing **of this pipe only** until a specific event occurs or while a specific condition exists. See Wait Events/Conditions, Device Programming Interface in *MI Functions*. Only one event/condition can be specified. Specifying multiple events is UNDEFINED. 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. 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	
Events must be unmasked in the Display Engine Render Response Mask Register (DE RRMR 0x44050) prior to waiting for them with a MI_WAIT_FOR_EVENT command, or in the case of flips or scanlines, prior to starting the flip or loading the scanline. Unmasked events will wake command streamer as they occur, so for improved power savings it is recommended to only unmask events that are required. Programming the DE RRMR register can be done through MMIO or a LOAD_REGISTER_IMMEDIATE command.	

Programming Notes	Project
Software must always program MI_NOOP command with “Identification Number Register Write Enable” set following MI_WAIT_FOR_EVENT command to avoid know HW issue.	

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	**Display Pipe C Horizontal Blank Wait Enable**

MI_WAIT_FOR_EVENT

<table>
<tr><td></td><td></td><td><table><tr><td>Format:</td><td>Enable</td></tr></table>This field enables a wait until the start of next Display Pipe C Horizontal Blank event occurs. This event is described as the start of the next Display C Horizontal blank period. Note that this can cause a wait for up to a line.</td></tr>
<tr><td></td><td>21</td><td><b>Display Pipe C Vertical Blank Wait Enable</b><table><tr><td></td><td></td></tr><tr><td>Format:</td><td>Enable</td></tr></table>This field enables a wait until the next Display Pipe C Vertical Blank event occurs. This event is described as the start of the next Display C vertical blank period. Note that this can cause a wait for up to an entire refresh period.</td></tr>
<tr><td></td><td>20</td><td><b>Display Sprite C Flip Pending Wait Enable</b><table><tr><td></td><td></td></tr><tr><td>Format:</td><td>Enable</td></tr></table>This field enables a wait for the duration of a Display Sprite C Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).</td></tr>
<tr><td></td><td>19:16</td><td><b>Condition Code Wait Select</b><table><tr><td></td><td></td></tr></table>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.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>Not enabled</td><td>Condition Code Wait Not Enabled</td></tr><tr><td>1h-5h</td><td>Enable</td><td>Condition Code Select Enabled; selects one of 5 codes, 0 – 4</td></tr><tr><td>6h-15h</td><td>Reserved</td><td></td></tr></table><table><tr><th>Programming Notes</th></tr><tr><td>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.</td></tr></table></td></tr>
<tr><td></td><td>15</td><td><b>Display Plane C Flip Pending Wait Enable</b><table><tr><td></td><td></td></tr><tr><td>Format:</td><td>Enable</td></tr></table>This field enables a wait for the duration of a Display Plane C “Flip Pending” condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).</td></tr>
<tr><td></td><td>14</td><td><b>Display Pipe C Scan Line Wait Enable</b><table><tr><td></td><td></td></tr><tr><td>Format:</td><td>Enable</td></tr></table>This field enables a wait while a Display Pipe C Scan Line condition exists. This condition is defined as the start of the scan line specified in the Pipe C Display Scan Line Count Range Compare Register.</td></tr>
<tr><td></td><td>13</td><td><b>Display Pipe B Horizontal Blank Wait Enable</b><table><tr><td></td><td></td></tr></table></td></tr>
</table>

MI_WAIT_FOR_EVENT

		Format: Enable This field enables a wait until the start of next Display Pipe B “Horizontal Blank” event occurs. This event is described as the start of the next Display B Horizontal blank period. Note that this can cause a wait for up to a line.
	12	**Reserved** Format: MBZ
	11	**Display Pipe B Vertical Blank Wait Enable** Format: U32 This field enables a wait until the next Display Pipe B “Vertical Blank” event occurs. This event is described 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.
	10	**Display Sprite B Flip Pending Wait Enable** 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).
	9	**Display Plane B Flip Pending Wait Enable** 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** Format: Enable This field enables a wait while a Display Pipe B Scan Line condition exists. This condition is defined as the start of the scan line specified in the Pipe B Display Scan Line Count Range Compare Register.
	7:6	**Reserved** Format: MBZ
	5	**Display Pipe A Horizontal Blank Wait Enable** Format: U32 This field enables a wait until the start of next Display Pipe A Horizontal Blank event occurs. This event is described 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** Format: MBZ
	3	**Display Pipe A Vertical Blank Wait Enable** Format: U32 This field enables a wait until the next Display Pipe A “Vertical Blank” event occurs. This event is described 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.

MI_WAIT_FOR_EVENT

	2	**Display Sprite A Flip Pending Wait Enable** 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).
	1	**Display Plane A Flip Pending Wait Enable** 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).
	0	**Display Pipe A Scan Line Wait Enable** Format: Enable This field enables a wait while a Display Pipe A “Scan Line” condition exists. This condition is defined as the start of the scan line specified in the Pipe A Display Scan Line Count Range Compare Register.

1.2.24 MI_LOAD_REGISTER_MEM

MI_LOAD_REGISTER_MEM

Source:	RenderCS
Length Bias:	2

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

Programming Notes

The command temporarily halts commands that will cause cycles down the 3D pipeline.

This command should not be used within a non-privilege batch buffer to access global virtual space, doing so will be treated as privilege access violation. Refer "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation.

This command is not allowed to update the privilege register range when executed from a non-privilege batch buffer.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 29h MI_LOAD_REGISTER_MEM Format: OpCode
	22	**Use Global GTT** This bit if set when executing from a non-privileged batch buffer will be treated as privilege access

<table>
<tr><th colspan="3">MI_LOAD_REGISTER_MEM</th></tr>
<tr><td></td><td></td><td>violation. It is allowed for this bit to be clear when executing this command from a privileged (secure) batch buffer or ring buffer.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Per Process Graphics Address</td><td></td></tr>
<tr><td>1h</td><td>Global Graphics Address</td><td>This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.</td></tr>
</table></td></tr>
<tr><td></td><td>21</td><td><b>Async Mode Enable</b>
<table>
<tr><th>Description</th><th>Project</th></tr>
<tr><td>If this bit is set then the command stream will not wait for completion of this command before executing the next command. Please refer to the LOAD_INDIRECT and Predicate registers for usage of this bit.</td><td></td></tr>
<tr><td></td><td></td></tr>
</table></td></tr>
<tr><td></td><td>20:8</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>7:0</td><td><b>DWord Length</b>
<table>
<tr><td>Default Value:</td><td>1h</td></tr>
<tr><td>Format:</td><td>=n Total Length - 2. Excludes DWord (0,1).</td></tr>
</table></td></tr>
<tr><td>1</td><td>31:26</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>25:2</td><td><b>Register Address</b>
<table>
<tr><td>Format:</td><td>MMIOAddress[22:2]MMIO_Register</td></tr>
</table>
This field specifies Bits 25:2 of the Register offset the DWord will be written to. As the register address must be DWord-aligned, Bits 1:0 of that address MBZ.</td></tr>
<tr><td></td><td>1:0</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td>2</td><td>31:2</td><td><b>Memory Address</b>
<table>
<tr><td>Format:</td><td>GraphicsAddress[31:2]MMIO_Register</td></tr>
</table>
This field specifies the address of the memory location where the register value specified in the DWord above will read from. The address specifies the DWord location of the data.

Range = GraphicsVirtualAddress[31:2] for a DWord register</td></tr>
<tr><td></td><td>1:0</td><td><b>Reserved</b>
<table>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
</table>

1.2.25 MI_URB_CLEAR

MI_URB_CLEAR

MI_URB_CLEAR

Source:	RenderCS
Length Bias:	2

The MI_URB_CLEAR command allows SW to clear (write zero) to a section in the URB.

Programming Notes

- The command temporarily halts command execution.
- This command is part of context save/restore. Only the last instance will be part of context.
- This command requires the 3D pipeline to be flushed before execution.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 19h MI_URB_CLEAR Format: OpCode
	22:8	**Reserved** Format: MBZ
	7:0	**DWord Length** Default Value: 0h Format: =n Total Length - 2. Excludes DWord (0,1).
1	31:30	**Reserved** Format: MBZ
	29	**Reserved** Format: MBZ
	28:16	**URB Clear Length** This field specifies the number of 256b entries in the URB to be cleared to zero. **Value** / **Name** [0,8191]
	15	**Reserved** Format: MBZ
	14	**Reserved** Format: MBZ
	13:0	**URB Address** Format: URBAddress[18:5] 256b aligned This field specifies Bits 18:5 of the URB Address

1.2.26 MI_PREDICATE

The MI_PREDICATE command is used to control the Predicate state bit, which in turn can be used to enable/disable the processing of 3DPRIMITIVE commands.

<table>
<tr><th colspan="3">MI_PREDICATE</th></tr>
<tr><td colspan="3"><table><tr><td>Source:</td><td>RenderCS</td></tr><tr><td>Length Bias:</td><td>1</td></tr></table></td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td><b>Command Type</b><table><tr><td>Default Value:</td><td>0h MI_COMMAND</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>28:23</td><td><b>MI Command Opcode</b><table><tr><td>Default Value:</td><td>0Ch MI_PREDICATE</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td></td><td>22:8</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:6</td><td><b>Load Operation</b>
This field controls if/how the Predicate state bit is modified.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>LOADOP_KEEP</td><td>The Predicate state bit is unmodified.</td></tr><tr><td>1h</td><td>Reserved</td><td></td></tr><tr><td>2h</td><td>LOADOP_LOAD</td><td>The Predicate state bit is loaded with the combine operation result.</td></tr><tr><td>3h</td><td>LOADOP_LOADINV</td><td>The Predicate state bit is loaded with the inverted combine operation result.</td></tr></table></td></tr>
<tr><td></td><td>5</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>4:3</td><td><b>Combine Operation</b>
This field controls if/how the result of the compare operation is combined with the current Predicate state bit.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>COMBINEOP_SET</td><td>The combine operation output the compare result unmodified.</td></tr><tr><td>1h</td><td>COMBINEOP_AND</td><td>The combine operation outputs the AND of the compare result and the current Predicate state bit.</td></tr><tr><td>2h</td><td>COMBINEOP_OR</td><td>The combine operation outputs the OR of the compare result and the current Predicate state bit.</td></tr><tr><td>3h</td><td>COMBINEOP_XOR</td><td>The combine operation outputs the XOR of the compare result and the current Predicate state bit.</td></tr></table></td></tr>
<tr><td></td><td>2</td><td><b>Reserved</b><table><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>1:0</td><td><b>Compare Operation</b>
This field controls how Data DWord 0 and Data DWord 1 fields are used to generate a compare operation result and possibly modify the PredicateData register.<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>COMPAREOP_TRUE</td><td>The compare operation outputs TRUE. The PredicateData register is unmodified.</td></tr></table></td></tr>
</table>

MI_PREDICATE

		1h	COMPAREOP_FALSE	The compare operation outputs FALSE. The PredicateData register is unmodified.
		2h	COMPAREOP_SRCS_EQUAL	(MItemp0 – MItemp1) is computed and loaded into the PredicateData register. The compare operation outputs (MItemp0 == MItemp1).
		3h	COMPAREOP_DELTAS_EQUAL	(MItemp0 – MItemp1) is computed and compared to the PredicateData register. If the values are equal, the compare operation outputs TRUE, otherwise it outputs FALSE. The PredicateData register is unmodified.

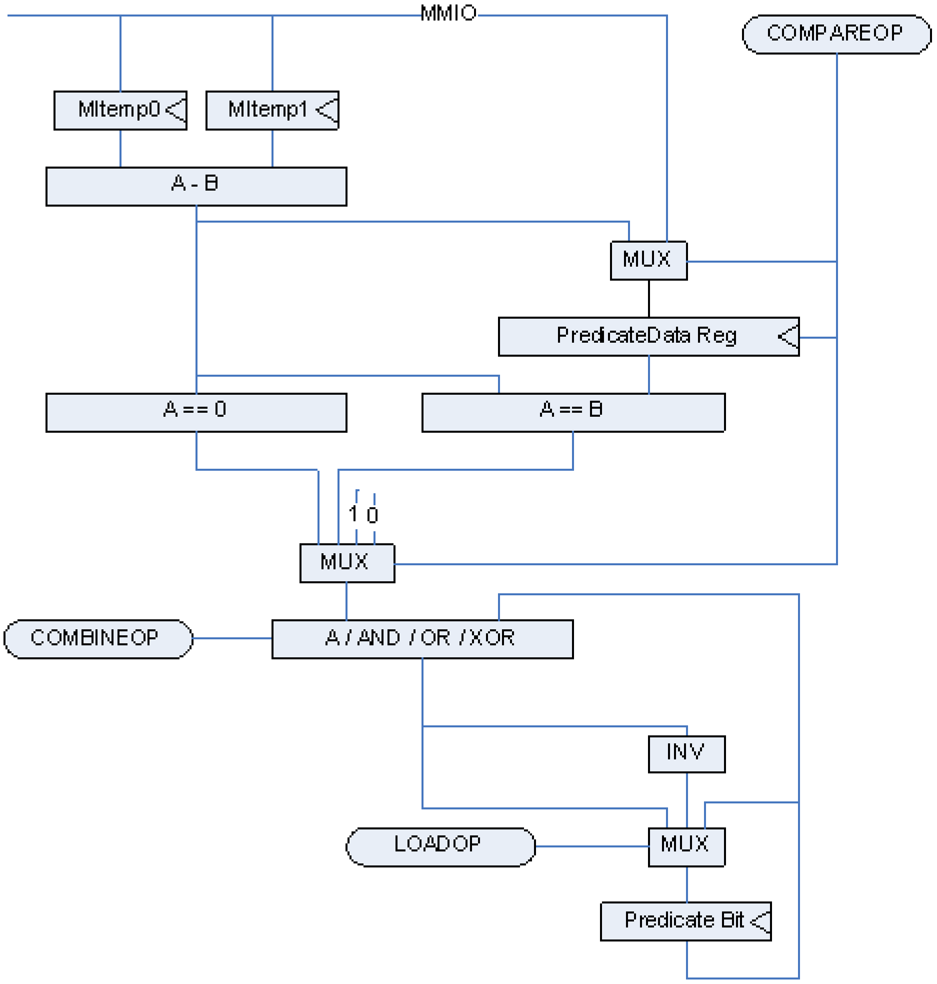
1.2.26.1 Predicated Rendering Support in HW

DX10 defines predicated rendering, where sequences of rendering commands can be discarded based on the result of a previous predicate test. A new state bit, Predicate, has been added to the command stream. In addition, a PredicateEnable bit is added to 3DPRIMITIVE. When the PredicateEnable bit is set, the command is ignored if the Predicate state bit is set.

A new command, MI_PREDICATE, is added. It contains several control fields which specify how the Predicate bit is generated.

Refer to the diagram below and the command description for details.

MI_PREDICATE Function



MI_LOAD_REGISTER_MEM commands can be used to load the MItemp0, MItemp1 and PredicateData registers prior to MI_PREDIATE. In order to ensure the memory sources of the MI_LOAD_REGISTER_MEM commands are coherent with previous 3D_PIPECONTROL store-dword operations, software can use the new **Pipe Control Flush Enable** bit in the PIPE_CONTROL command.

1.2.27 MI_TOPOLOGY_FILTER

MI_TOPOLOGY_FILTER	
Source:	RenderCS
Length Bias:	1
This command is used to specify a specific 3DPrimType value, where the CS will ignore all 3DPRIMITIVE commands that do no have a matching 3DPrimType. This primitive culling is optional (turned off by using this command with a Topology Filter Value of 0). **This command is specific to the Render command stream only.**	

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

MI_TOPOLOGY_FILTER

		Format: OpCode
	28:23	**MI Command Opcode** Default Value: 0Dh MI_TOPOLOGY_FILTER Format: OpCode
	22:6	**Reserved** Format: MBZ
	5:0	**Topology Filter Value** Format: 3D_PrimTopoType When non-zero, the CS will discard all 3DPRIMITIVE commands which do not match the specified 3DPrimTopologyType. When zero, no filtering is performed (normal operation).

Revision History

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

§§