Intel® Open Source HD Graphics Programmers' Reference Manual (PRM)

Volume 3: GPU Overview

For the 2014 Intel Atom™ Processors, Celeron™ Processors, and Pentium™ Processors based on the "BayTrail" Platform (Valleyview graphics)

Creative Commons License

You are free to Share — to copy, distribute, display, and perform the work

Under the following conditions:

Attribution. You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

No Derivative Works. You may not alter, transform, or build upon this work

Notices and Disclaimers

Table of Contents

Introduction

The integrated graphics component, specifically called the Graphics Processing Unit, or GPU, resides on the same chip die as the Central Processing Unit, or CPU, and communicates with the CPU via the on-chip bus, with internal memory and with output device(s). As Intel GPUs have evolved, they now occupy a significant percentage of space on the chip, and provide customers with high performance and low-power graphics processing, eliminating the need to purchase a separate video card for most users.

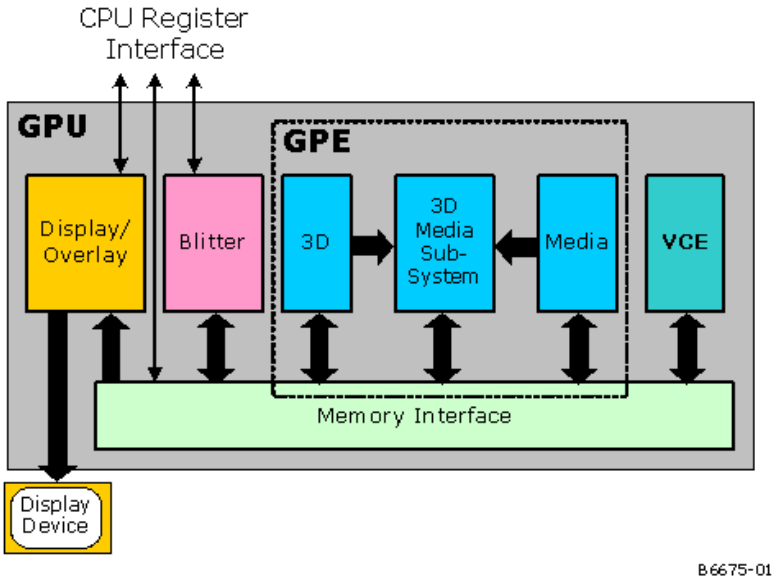
This PRM provides detailed narrative and reference information required by graphics device driver engineers and graphics API-level programmers to take advantage of the sophisticated architecture and programmability of the GPU.

Graphics Processing Unit (GPU)

The Graphics Processing Unit is controlled by the CPU through a direct interface of memory-mapped IO registers, and indirectly by parsing commands that the CPU has placed in memory. The Display interface and Blitter (**bl**ock **i**mage **t**ransfer**er**) are controlled primarily by direct CPU register addresses, while the 3D and Media pipelines and the parallel Video Codec Engine (VCE) are controlled primarily through instruction lists in memory.

The subsystem contains an array of cores, or execution units, with a number of "shared functions", which receive and process messages at the request of programs running on the cores. The shared functions perform critical tasks, such as sampling textures and updating the render target (usually the frame buffer). The cores themselves are described by an instruction set architecture, or ISA.

Block Diagram of the GPU



Command Stream (CS) Unit

The Command Stream (CS) unit manages the use of the 3D and Media pipelines; it performs switching between pipelines and forwarding command streams to the currently active pipeline. It manages allocation of the URB and helps support the Constant URB Entry (CURBE) function.

3D Pipeline

The 3D Pipeline provides specialized 3D primitive processing functions. These functions are provided by a pipeline of "fixed function" stages (units) and GEN threads spawned by these units. See *3D Pipeline Overview*.

Media Pipeline

The Media pipeline provides both specialized media-related processing functions and the ability to perform more general ("generic") functionality. These Media-specific functions are provided by a Video Front End (VFE) unit. A Thread Spawner (TS) unit is utilized to spawn GEN threads requested by the VFE unit, or as required when the pipeline is used for general processing. See *Media Pipeline Overview*.

Thread Dispatching

When the 3D and Media pipelines send requests for thread initiation to the Subsystem, the thread Dispatcher receives the requests. The dispatcher performs such tasks as arbitrating between concurrent requests, assigning requested threads to hardware threads on EUs, allocating register space in each EU among multiple threads, and initializing a thread's registers with data from the fixed functions and from the URB. This operation is largely transparent to software.

Execution Units (EUs)

While the number of EU cores in the subsystem is almost entirely transparent to the programming model, this parameter does come into play because the amount of scratch space required is a function of the number of EUs times Threads/EU. VLV has 4 EUs with 8 Threads/EU as shown below:

Device	# of EUs	#Threads/EU
[VLV]	4	8

Shared Functions

Shared functions are hardware units which serve to provide specialized supplemental functionality for the EUs. A shared function is implemented where the demand for a given specialized function is insufficient to justify the costs on a per-EU basis. Instead a single instantiation of that specialized function is implemented as a stand-alone entity outside the EUs and shared among the EUs.

Invocation of the shared functionality is performed via a communication mechanism called a message. A message is a small self-contained packet of information created by a kernel and directed to a specific shared function.

The message construction and delivery mechanisms are general in their definition and capable of supporting a wide variety of shared functions.

Fixed and Shared Function IDs

The following table lists the assignments (encodings) of the Shared Function and Fixed Function IDs used within the GPE. A Shared Function is a valid target of a message initiated via a 'send' instruction. A Fixed Function is an identifiable unit of the 3D or Media pipeline. Note that the Thread Spawner is both a Shared Function and Fixed Function.

Function IDs

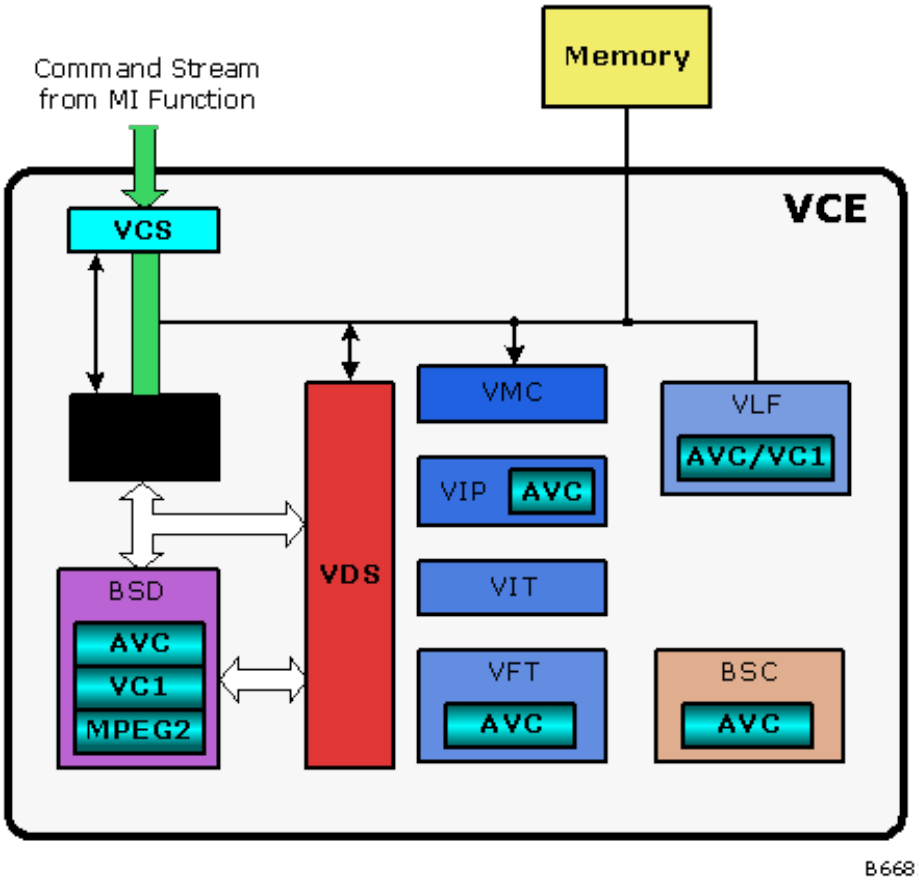
ID[3:0]	SFID	Shared Function	FFID	Fixed Function
0x0	SFID_NULL	Null	FFID_NULL	Null
0x1	Reserved	---	**Reserved**	---
0x2	SFID_SAMPLER	Sampler	**Reserved**	---
0x3	SFID_GATEWAY	Message Gateway	**Reserved**	---
0x4	SFID_DP_SAMPLER	Sampler Cache Data Port	FFID_HS	Hull Shader
0x5	SFID_DP_RC	Render Cache Data Port	FFID_DS	Domain Shader
0x6	SFID_URB	URB	**Reserved**	---
0x7	SFID_SPAWNER	Thread Spawner	FFID_SPAWNER	Thread Spawner
0x8	SFID_VME	Video Motion Estimation	FFID_VFE	Video Front End
0x9	SFID_DP_CC	Constant Cache Data Port	FFID_VS	Vertex Shader
0xA	SFID_DP_DC	Data Cache Data Port	FFID_CS	Command Stream
0xB	SFID_PI	Pixel Interpolator	FFID_VF	Vertex Fetch
0xC	Reserved	---	FFID_GS	Geometry Shader
0xD	Reserved	---	FFID_CLIP	Clipper Unit
0xE	Reserved	---	FFID_SF	Strip/Fan Unit
0xF	Reserved	---	FFID_WM	Windower/Masker Unit

Video Codec Engine

The parallel Video Codec Engine (VCE) is a fixed function video decoder and encoder engine. It is also referred to as the multi-format codec (MFX) engine, as a unified fixed function pipeline is implemented to support multiple video coding standards such as MPEG2, VC1, and AVC:

- VCS – VCE Command Streamer unit (also referred to as BCS)
- BSD – Bitstream Decoder unit
- VDS – Video Dispatcher unit
- VMC – Video Motion Compensation unit
- VIP – Video Intra Prediction unit
- VIT – Video Inverse Transform unit
- VLF – Video Loop Filter unit
- VFT – Video Forward Transform unit (encoder only)
- BSC – Bitstream Encoder unit (encoder only)

VCE Diagram



AVC BSD	VC1 BSD	AVC Decode	VC1 Decode	MPEG2 Decode	AVC Encode
No	No	Yes	Yes	Yes	Yes

Register Address Maps

This section discusses the following topics:

- Graphics Register Address Map
- VGA and Extended VGA Register Address Map

Graphics Register Address Map

This section provides information about the following topics:

- Memory and I/O Space Registers
- Graphics Register Memory Address Map

Graphics Register Memory Address Map

All graphics device registers are directly accessible via memory-mapped I/O and indirectly accessible via the MMIO_INDEX and MMIO_DATA I/O registers. In addition, the VGA and Extended VGA registers are I/O mapped.

Display registers are documented in the VGA and Display Registers volumes.

Table: Memory-Mapped Registers

Address Offset	Symbol	Register Name	Access
Reserved (1000h 1FFFh)			
01000h 01FFFh		**Reserved**	
Primary CS Instruction and Interrupt Control Registers (02000h 02FFFh)			
02000h 0201Fh		**Reserved**	
02020h 02023h	PGTBL_CTL	Page Table Control Register	R/W
02024h 02027h			RO
02028h–0202Bh	EXCC	Execute Condition Code Register	R/W,RO
0202Ch–0202Fh		**Reserved**	
02030h–02033h	PRB0_TAIL	Primary Ring Buffer 0 Tail Register	R/W
02034h–02037h	PRB0_HEAD	Primary Ring Buffer 0 Head Register	R/W
02038h–0203Bh	PRB0_STARTsted	Primary Ring Buffer 0 Start Register	R/W
0203Ch–0203Fh	PRB0_CTL	Primary Ring Buffer 0 Control Register	R/W
02040h–0205Fh		**Reserved**	
02080h–02083h	HWS_PGA	Hardware Status Page Address Register	R/W
02084h–02087h		**Reserved**	
0208Dh–02093h		**Reserved**	
02094h–02097h	NOPID	NOP Identification Register	RO
02098h 0209Bh	HWSTAM	Hardware Status Mask Register	R/W

Address Offset	Symbol	Register Name	Access
Reserved (1000h 1FFFh)			
0209Ch–0209Fh	MI_MODE	Mode Register for Software Interface	R/W
020A0h 020A3h	IER	Interrupt Enable Register	R/W
020A4h 020A7h	IIR	Interrupt Identity Register	R/WC
020A8h 020ABh	IMR	Interrupt Mask Register	R/W
020ACh 020AFh	ISR	Interrupt Status Register	RO
020B0h 020B3h	EIR	Error Identity Register	R/WC
020B4h 020B7h	EMR	Error Mask Register	R/W
020B8h 020BBh	ESR	Error Status Register	RO
020BCh 020BFh		**Reserved**	
020C0h–020C3h	INSTPM	Instruction Parser Mode Register (SAVED/RESTORED)	R/W
020CCh–020DFh		**Reserved**	
020E4h 020E7h	MI_ARB_STATE	Memory Interface Arbitration State Register (SAVED/RESTORED)	R/W
020E8h 020FBh		**Reserved**	
02100h–0210Fh		**Reserved**	
02114h–0211Fh		**Reserved**	
02120h–02123h			R/W
02124h 02127h			R/W
02128h–02133h		**Reserved**	
02134h–02137h	UHPTR	Pending Head Pointer Register	R/W
02138h–0213Fh		**Reserved**	
02140h–02147h	BB_ADDR	Batch Buffer Current Address	RO
0214Ch–0216Fh		**Reserved**	
02170h–02177h	GFX_FLSH_CNTL	Graphics Flush Control	R/W
02180h 02183h	CCID0	Current Context ID 0 (assoc w/ PRB0)	R/W
02184h 0218Fh		**Reserved**	
02194h 0219Fh		**Reserved**	
021A0h 021A3h			R/W
021A4h 021A7h			R/W
021A8h-021CFh		**Reserved**	
021D0h-021D3h			R/W
021D4h-021FFh		**Reserved**	

Address Offset	Symbol	Register Name	Access
Reserved (1000h 1FFFh)			
02214h 0230Fh		**Reserved**	
02310h-02317h	IA_VERTICES_COUNT	Reported Vertices Counter	R/W
02318h-0231Fh	IA_PRIMITVES_COUNT	Reported Vertex Fetch Output Primitives Counter	R/W
02320h-02327h	VS_INVOCATION_COUNT	Reported Vertex Shader Thread Invocation Counter	R/W
02328h-0232Fh	GS_INVOCATION_COUNT	Reported Geometry Shader Thread Invocation Counter	R/W
02330h-02337h	GS_PRIMITIVES_COUNT	Reported Geometry Shader Output Primitives Counter	R/W
02338h-0233Fh	CL_INVOCATION_COUNT	Reported Clipper Thread Invocation Counter	R/W
02340h-02347h	CL_PRIMITIVES_COUNT	Reported Clipper Output Primitives Counter	R/W
02348h-0234Fh	PS_INVOCATION_COUNT	Reported Pixels Shaded Invocation Counter	R/W
02358–0235Fh	TIMESTAMP	Reported Timestamp Count	R/W
02360–02367h			
02368h 0236Fh		**Reserved**	
02388h 0244Fh		**Reserved**	
02450h 02453h			R/W
02454h 0246Fh		**Reserved**	
02470h 02473h			R/W
02474h 024FFh		**Reserved**	
MFC Status Registers (012400h 012444h)			
		Reserved	
12400h	AVD Error flags	AVD Internal Error flags	RW
12404h	AVD Error counter	AVD Error Counter	RW
12408h	MFC_BITSTREAM_BYTECOUNT_SLICE	Bitstream Output Byte Count Register per Slice	RO
1240Ch	MFC_BITSTREAM_SE_BITCOUNT_SLICE	Bitstream Output Bit Count for the last Syntax Element Register	RO
12410h	MFC_AVC_CABAC_INSERTION_COUNT	Bitstream Output CABAC Insertion Count Register	RO
12414h	MFC_AVC_MINSIZE_PADDING_COUNT	Bitstream Output Minimal Size Padding Count Register	RO
12418h	MFC_IMAGE_STATUS_MASK	Image Coding Status Mask Register	RO
1241Ch	MFC_IMAGE_STATUS_CONTROL	Image Coding Status Control Register	RO
12420h	MFC_BITSTREAM_BYTECOUNT_FRAME	Total Bitstream Output Byte Count register per Frame	RO

Address Offset	Symbol	Register Name	Access
Reserved (1000h 1FFFh)			
12424h	MFC_BITSTREAM_SE_BITCOUNT_FRAME	Bitstream Output total Byte Count for syntax eements (total byes of MB data from SEC per frame)	RO
12428h	MFC_AVC_CABAC_BIN_COUNT_FRAME	Bitstream Output total bin count per frame	RO
VSC Registers (05000h – 05FFFh)			
05000h-05003h		FMD Variance 0 for Stream 0	
05004h-05007h		FMD Variance 1 for Stream 0	
05008h-0500Bh		FMD Variance 2 for Stream 0	
0500Ch-0500Fh		FMD Variance 3 for Stream 0	
05010h-05013h		FMD Variance 4 for Stream 0	
05014h-05017h		FMD Variance 5 for Stream 0	
05018h-0501Bh		FMD Variance 6 for Stream 0	
0501Ch-0501Fh		FMD Variance 7 for Stream 0	
05020h-05023h		FMD Variance 8 for Stream 0	
05024h-05027h		FMD Variance 9 for Stream 0	
05028h-0502Bh		FMD Variance 10 for Stream 0	
0502Ch-0502Fh		GNE Sum for Stream0	
05030h-05033h		Number of Valid GNE Blocks Strm0	
05034h-05037h		FMD Variance 0 for Stream 1	
05038h-0503Bh		FMD Variance 1 for Stream 1	
0503Ch-0503Fh		FMD Variance 2 for Stream 1	
05040h-05043h		FMD Variance 3 for Stream 1	
05044h-05047h		FMD Variance 4 for Stream 1	
05048h-0504Bh		FMD Variance 5 for Stream 1	
0504Ch-0504Fh		FMD Variance 6 for Stream 1	
05050h-05053h		FMD Variance 7 for Stream 1	
05054h-05057h		FMD Variance 8 for Stream 1	
05058h-0505Bh		FMD Variance 9 for Stream 1	
0505Ch-0505Fh		FMD Variance 10 for Stream 1	
05060h-05063h		GNE Sum for Stream 1	
05064h-05067h		Number of Valid GNE Blocks Strm1	
05068h-0506Fh		**Reserved**	
VSC Registers			
05070h-05073h		Ymax (bits 25:16]), Ymin (bits[9:0]), other bits zero	
05074h-05077h		Number of skin pixels (bits [20:0], other bits zero)	

Address Offset	Symbol	Register Name	Access
Reserved (1000h 1FFFh)			
05078h-0507Fh		**Reserved**	
05080h-05081h		ACE Histogram, bin 0	
05082h-05083h		ACE Histogram, bin 1	
…		ACE Histogram, bins 2 - 126	
0517Eh-0517Fh		ACE Histogram, bin 127	
Clock Control and Power Management Registers (06000h 06FFFh)			
06000h 06003h	VGA0	VGA 0 Divisor	R/W
06004h 06007h	VGA1	VGA 1 Divisor	R/W
06008h 0600Fh		**Reserved**	
06010h 06013h	VGA_PD	VGA Post Divisor Select	R/W
06014h 06017h	DPLLA_CTRL	Display PLL A Control	R/W
06018h 0601Bh	DPLLB_CTRL	Display PLL B Control	R/W
06024h 0603Fh		**Reserved**	
06040h 06043h	FPA0	DPLL A Divisor 0	R/W
06044h 06047h	FPA1	DPLL A Divisor 1	R/W
06048h 0604Bh	FPB0	DPLL B Divisor 0	R/W
0604Ch 0604Fh	FPB1	DPLL B Divisor 1	R/W
06050h 0606Bh		**Reserved**	
0606Ch–0606Fh			R/W
06070h 06103h		**Reserved**	
06104h 06107h	D_STATE	D State Function Control	R/W
06108h 061FFh		**Reserved**	
06200h 06203h	DSPCLK_GATE_D	Clock Gating Disable for Display Register	R/W
06204h 06207h	RENCLK_GATE_D1	Clock Gating Disable for Render Register I	R/W
06208h 0620Bh	RENDCLK_GATE_D2	Clock Gating Disable for Render Register II	
06214h–06125h		**Reserved**	
06216h 06FFFh		**Reserved**	
GFX MMIO – MCHBAR Aperture (10000h-13FFFh)			
10000h-13FFFh		MCHBAR Aperture	R/W
Reserved (14000h-2FFFFh)			
14000h-2FFFFh		**Reserved**	

VGA and Extended VGA Register Map

For I/O locations, the value in the address column represents the register I/O address. For memory mapped locations, this address is an offset from the base address programmed in the MMADR register.

VGA and Extended VGA I/O and Memory Register Map

Address	Register Name (Read)	Register Name (Write)
2D Registers		
3B0h–3B3h	Reserved	Reserved
3B4h	VGA CRTC Index (CRX) (monochrome)	VGA CRTC Index (CRX) (monochrome)
3B5h	VGA CRTC Data (monochrome)	VGA CRTC Data (monochrome)
3B6h–3B9h	Reserved	Reserved
3Bah	VGA Status Register (ST01)	VGA Feature Control Register (FCR)
3BBh–3BFh	Reserved	Reserved
3C0h	VGA Attribute Controller Index (ARX)	VGA Attribute Controller Index (ARX)/ VGA Attribute Controller Data (alternating writes select ARX or write ARxx Data)
3C1h	VGA Attribute Controller Data (read ARxx data)	Reserved
3C2h	VGA Feature Read Register (ST00)	VGA Miscellaneous Output Register (MSR)
3C3h	Reserved	Reserved
3C4h	VGA Sequencer Index (SRX)	VGA Sequencer Index (SRX)
3C5h	VGA Sequencer Data (SRxx)	VGA Sequencer Data (SRxx)
3C6h	VGA Color Palette Mask (DACMASK)	VGA Color Palette Mask (DACMASK)
3C7h	VGA Color Palette State (DACSTATE)	VGA Color Palette Read Mode Index (DACRX)
3C8h	VGA Color Palette Write Mode Index (DACWX)	VGA Color Palette Write Mode Index (DACWX)
3C9h	VGA Color Palette Data (DACDATA)	VGA Color Palette Data (DACDATA)
3CAh	VGA Feature Control Register (FCR)	Reserved
3CBh	Reserved	Reserved
3CCh	VGA Miscellaneous Output Register (MSR)	Reserved
3CDh	Reserved	Reserved
3CEh	VGA Graphics Controller Index (GRX)	VGA Graphics Controller Index (GRX)
3CFh	VGA Graphics Controller Data (GRxx)	VGA Graphics Controller Data (GRxx)
3D0h–3D1h	Reserved	Reserved
2D Registers		
3D4h	VGA CRTC Index (CRX)	VGA CRTC Index (CRX)
3D5h	VGA CRTC Data (CRxx)	VGA CRTC Data (CRxx)
System Configuration Registers		

Address	Register Name (Read)	Register Name (Write)
2D Registers		
3D6h	GFX/2D Configurations Extensions Index (XRX)	GFX/2D Configurations Extensions Index (XRX)
3D7h	GFX/2D Configurations Extensions Data (XRxx)	GFX/2D Configurations Extensions Data (XRxx)
2D Registers		
3D8h–3D9h	Reserved	Reserved
3DAh	VGA Status Register (ST01)	VGA Feature Control Register (FCR)
3DBh–3DFh	Reserved	Reserved

Indirect VGA and Extended VGA Register Indices

The registers listed in this section are indirectly accessed by programming an index value into the appropriate SRX, GRX, ARX, or CRX register. The index and data register address locations are listed in the previous section. Additional details concerning the indirect access mechanism are provided in the *VGA and Extended VGA Register Description* Chapter (see SRxx, GRxx, ARxx or CRxx sections).

2D Sequence Registers (3C4h / 3C5h)

Index	Sym	Description
00h	SR00	Sequencer Reset
01h	SR01	Clocking Mode
02h	SR02	Plane / Map Mask
03h	SR03	Character Font
04h	SR04	Memory Mode
07h	SR07	Horizontal Character Counter Reset

2D Graphics Controller Registers (3CEh / 3CFh)

Index	Sym	Register Name
00h	GR00	Set / Reset
01h	GR01	Enable Set / Reset
02h	GR02	Color Compare
03h	GR03	Data Rotate
04h	GR04	Read Plane Select
05h	GR05	Graphics Mode
06h	GR06	Miscellaneous
07h	GR07	Color Don’t Care
08h	GR08	Bit Mask
10h	GR10	Address Mapping
11h	GR11	Page Selector
18h	GR18	Software Flags

2D Attribute Controller Registers (3C0h / 3C1h)

Index	Sym	Register Name
00h	AR00	Palette Register 0
01h	AR01	Palette Register 1
02h	AR02	Palette Register 2
03h	AR03	Palette Register 3
04h	AR04	Palette Register 4
05h	AR05	Palette Register 5
06h	AR06	Palette Register 6
07h	AR07	Palette Register 7
08h	AR08	Palette Register 8
09h	AR09	Palette Register 9
0Ah	AR0A	Palette Register A
0Bh	AR0B	Palette Register B
0Ch	AR0C	Palette Register C
0Dh	AR0D	Palette Register D
0Eh	AR0E	Palette Register E
0Fh	AR0F	Palette Register F
10h	AR10	Mode Control
11h	AR11	Overscan Color
12h	AR12	Memory Plane Enable
13h	AR13	Horizontal Pixel Panning
14h	AR14	Color Select

2D CRT Controller Registers (3B4h / 3D4h / 3B5h / 3D5h)

Index	Sym	Register Name
00h	CR00	Horizontal Total
01h	CR01	Horizontal Display Enable End
02h	CR02	Horizontal Blanking Start
03h	CR03	Horizontal Blanking End
04h	CR04	Horizontal Sync Start
05h	CR05	Horizontal Sync End
06h	CR06	Vertical Total
07h	CR07	Overflow
08h	CR08	Preset Row Scan
09h	CR09	Maximum Scan Line
0Ah	CR0A	Text Cursor Start
0Bh	CR0B	Text Cursor End
0Ch	CR0C	Start Address High
0Dh	CR0D	Start Address Low
0Eh	CR0E	Text Cursor Location High
0Fh	CR0F	Text Cursor Location Low
10h	CR10	Vertical Sync Start
11h	CR11	Vertical Sync End
12h	CR12	Vertical Display Enable End
13h	CR13	Offset
14h	CR14	Underline Location
15h	CR15	Vertical Blanking Start
16h	CR16	Vertical Blanking End
17h	CR17	CRT Mode
18h	CR18	Line Compare
22h	CR22	Memory Read Latch Data

Memory Object Overview

Any memory data accessed by the device is considered part of a *memory object* of some memory object type.

The following table lists the various memory objects types and an indication of their role in the system.

Memory Object Type	Role
Graphics Translation Table (GTT)	Contains PTEs used to translate "graphics addresses" into physical memory addresses.
Hardware Status Page	Cached page of sysmem used to provide fast driver synchronization.
Logical Context Buffer	Memory areas used to store (save/restore) images of hardware rendering contexts. Logical contexts are referenced via a pointer to the corresponding Logical Context Buffer.
Ring Buffers	Buffers used to transfer (DMA) instruction data to the device. Primary means of controlling rendering operations.
Batch Buffers	Buffers of instructions invoked indirectly from Ring Buffers.
State Descriptors	Contains state information in a prescribed layout format to be read by hardware. Many different state descriptor formats are supported.
Vertex Buffers	Buffers of 3D vertex data indirectly referenced through "indexed" 3D primitive instructions.
VGA Buffer (Must be mapped UC on PCI)	Graphics memory buffer used to drive the display output while in legacy VGA mode.
Display Surface	Memory buffer used to display images on display devices.
Overlay Surface	Memory buffer used to display overlaid images on display devices.
Overlay Register, Filter Coefficients Buffer	Memory area used to provide double-buffer for Overlay register and filter coefficient loading.
Cursor Surface	Hardware cursor pattern in memory.
2D Render Source	Surface used as primary input to 2D rendering operations.
2D Render R-M-W Destination	2D rendering output surface that is read in order to be combined in the rendering function. Destination surfaces that accessed via this Read-Modify-Write mode have somewhat different restrictions than Write-Only Destination surfaces.
2D Render Write-Only Destination	2D rendering output surface that is written but not read by the 2D rendering function. Destination surfaces that accessed via a Write-Only mode have somewhat different restrictions than Read-Modify-Write Destination surfaces.
2D Monochrome Source	1 bpp surfaces used as inputs to 2D rendering after being converted to foreground/background colors.
2D Color Pattern	8x8 pixel array used to supply the "pattern" input to 2D rendering functions.
DIB	"Device Independent Bitmap" surface containing "logical" pixel values that are converted (via LUTs) to physical colors.
3D Color Buffer	Surface receiving color output of 3D rendering operations. May also be accessed via R-M-W (aka blending). Also referred to as a Render Target.

Memory Object Type	Role
3D Depth Buffer	Surface used to hold per-pixel depth and stencil values used in 3D rendering operations. Accessed via RMW.
3D Texture Map	Color surface (or collection of surfaces) which provide texture data in 3D rendering operations.
"Non-3D" Texture	Surface read by Texture Samplers, though not in normal 3D rendering operations (e.g., in video color conversion functions).
Motion Comp Surfaces	These are the Motion Comp reference pictures.
Motion Comp Correction Data Buffer	This is Motion Comp intra-coded or inter-coded correction data.

Hardware Status Page

The hardware status page is a naturally-aligned 4KB page residing in snooped system memory. This page exists primarily to allow the device to report status via PCI master writes – thereby allowing the driver to read/poll WB memory instead of UC reads of device registers or UC memory.

The address of this page is programmed via the HWS_PGA MI register. The definition of that register (in *Memory Interface Registers*) includes a description of the layout of the Hardware Status Page.

Instruction Ring Buffers

Instruction ring buffers are the memory areas used to pass instructions to the device. Refer to the Programming Interface chapter for a description of how these buffers are used to transport instructions.

The RINGBUF register sets (defined in Memory Interface Registers) are used to specify the ring buffer memory areas. The ring buffer must start on a 4KB boundary and be allocated in linear memory. The length of any one ring buffer is limited to 2MB.

Note that "indirect" 3D primitive instructions (those that access vertex buffers) must reside in the same memory space as the vertex buffers.

Instruction Batch Buffers

Instruction batch buffers are contiguous streams of instructions referenced via an MI_BATCH_BUFFER_START and related instructions (see Memory Interface Instructions, Programming Interface). They are used to transport instructions external to ring buffers.

Note that batch buffers should not be mapped to snooped SM (PCI) addresses. The device will treat these as MainMemory (MM) address, and therefore not snoop the CPU cache.

The batch buffer must be QWord aligned and a multiple of QWords in length. The ending address is the address of the last valid QWord in the buffer. The length of any single batch buffer is "virtually unlimited" (i.e., could theoretically be 4GB in length).

Logical Contexts

This section discusses the following topics:

- BSD Logical Context Data
- Copy Engine Logical Context Data

BSD Logical Render Context Address (LRCA)

This section discusses the following topics for BSD Logical Render Context Address:

- Overall Context Layout
- Register State Context
- The Per Process Hardware Status Page

Overall Context Layout

BSD effectively has no context. Switching from one task to another is accomplished by programming the UHPTR register with a new head pointer, then executing an MI_ARB_CHECK command. This will load the head pointer with the new value, "jumping" to the commands for the next task.

Register/State Context

DW Range	DW Count	Restore Inhibited	Power Context	Set Before Submitting Context?
00h	1	R	S/R	Yes
01h	1	R	S/R	Yes
02h	1	R	S/R	Yes
03h	1	NR	S/R	No
04h	1	NR	S/R	No
05h	1	R	S/R	Yes
06h	1	X	S/R	X
07h	1	R	S/R	Yes
08h	1	NR	S/R	Yes
09h	1	NR	S/R	Yes
0Ah	1	NR	S/R	Yes
0Bh	1	NR	S/R	Yes
0Ch	1	NR	S/R	No
0Dh	1	NR	S/R	No
0Eh	1	NR	S/R	No
0Fh	1	X	S/R	X

The Per-Process Hardware Status Page

The following table defines the layout of the Per-process Hardware Status Page:

DWord Offset	Description
(3FFh – 020h)	These locations can be used for general purpose via the MI_STORE_DATA_INDEX or MI_STORE_DATA_IMM instructions.
1F:5	**Reserved.**
4	**Ring Head Pointer Storage:** The contents of the Ring Buffer Head Pointer register (register DWord 1) are written to this location either as result of an MI_REPORT_HEAD instruction or as the result of an "automatic report" (see RINGBUF registers).
3:0	**Reserved.**

This page is designed to be read by SW in order to glean additional details about a context beyond what it can get from the context status.

Accesses to this page will automatically be treated as cacheable and snooped. It is therefore illegal to locate this page in any region where snooping is illegal (such as in stolen memory).

Copy Engine Logical Context Data

This section discusses the following topics for Copy Engine Logical Context Data:

- Overall Context Layout
- Register State Context
- The Per Process Hardware Status Page

Overall Context Layout

The video engine effectively has no context. Switching from one task to another only occurs when the head pointer equals the tail pointer and there is a new context ID received.

Register/State Context

DW Range	DW Count	Restore Inhibited	Power Context	Set Before Submitting Context?
00h	1	R	S/R	Yes
01h	1	R	S/R	Yes
02h	1	R	S/R	Yes
03h	1	NR	S/R	No
04h	1	NR	S/R	No
05h	1	R	S/R	Yes
06h	1	X	S/R	X

07h	1	R	S/R	Yes
08h	1	NR	S/R	Yes
09h	1	NR	S/R	Yes
0Ah	1	NR	S/R	Yes
0Bh	1	NR	S/R	Yes
0Ch	1	NR	S/R	No
0Dh	1	NR	S/R	No
0Eh	1	NR	S/R	No
0Fh	1	X	S/R	X

The Per-Process Hardware Status Page

The following table defines the layout of the Per-process Hardware Status Page:

DWord Offset	Description
(3FFh – 020h)	These locations can be used for general purpose via the MI_STORE_DATA_INDEX or MI_STORE_DATA_IMM instructions.
1F:5	**Reserved.**
4	**Ring Head Pointer Storage:** The contents of the Ring Buffer Head Pointer register (register DWord 1) are written to this location either as result of an MI_REPORT_HEAD instruction or as the result of an "automatic report" (see RINGBUF registers).
3:0	**Reserved.**

This page is designed to be read by SW in order to glean additional details about a context beyond what it can get from the context status.

Accesses to this page will automatically be treated as cacheable and snooped. It is therefore illegal to locate this page in any region where snooping is illegal (such as in stolen memory).

Memory Data Formats

This chapter describes the attributes associated with the memory-resident data objects operated on by the graphics pipeline. This includes object types, pixel formats, memory layouts, and rules/restrictions placed on the dimensions, physical memory location, pitch, alignment, etc. with respect to the specific operations performed on the objects.

Unsigned Normalized (UNORM)

An unsigned normalized value with *n* bits is interpreted as a value between 0.0 and 1.0. The minimum value (all 0's) is interpreted as 0.0, the maximum value (all 1's) is interpreted as 1.0. Values in between are equally spaced. For example, a 2-bit UNORM value would have the four values 0, 1/3, 2/3, and 1.

If the incoming value is interpreted as an n-bit integer, the interpreted value can be calculated by dividing the integer by 2^n-1.

Gamma Conversion (SRGB)

Gamma conversion is only supported on UNORM formats. If this flag is included in the surface format name, it indicates that a reverse gamma conversion is to be done after the source surface is read, and a forward gamma conversion is to be done before the destination surface is written.

Signed Normalized (SNORM)

A signed normalized value with *n* bits is interpreted as a value between -1.0 and +1.0. If the incoming value is interpreted as a 2's-complement n-bit signed integer, the interpreted value can be calculated by dividing the integer by 2^{n-1}-1. Note that the most negative value of -2^{n-1} will result in a value slightly smaller than -1.0. This value is clamped to -1.0, thus there are two representations of -1.0 in SNORM format.

Unsigned Integer (UINT/USCALED)

The UINT and USCALED formats interpret the source as an unsigned integer value with *n* bits with a range of 0 to 2^n-1.

The UINT formats copy the source value to the destination (zero-extending if required), keeping the value as an integer.

The USCALED formats convert the integer into the corresponding floating point value (e.g., 0x03 --> 3.0f). For 32-bit sources, the value is rounded to nearest even.

Signed Integer (SINT/SSCALED)

A signed integer value with *n* bits is interpreted as a 2's complement integer with a range of -2^{n-1} to $+2^{n-1}$-1.

The SINT formats copy the source value to the destination (sign-extending if required), keeping the value as an integer.

The SSCALED formats convert the integer into the corresponding floating point value (e.g., 0xFFFD --> -3.0f). For 32-bit sources, the value is rounded to nearest even.

Floating Point (FLOAT)

Refer to IEEE Standard 754 for Binary Floating-Point Arithmetic. The IA-32 Intel (R) Architecture Software Developer's Manual also describes floating point data types .

64-bit Floating Point

Bit	Description
63	**Sign (s)**
62:52	**Exponent (e)** Biased Exponent
51:0	**Fraction (f)** Does not include "hidden one"

The value of this data type is derived as:

- if e == b'11..11' and f != 0, then v is NaN regardless of s
- if e == b'11..11' and f == 0, then v = $(-1)^{s}$*infinity (signed infinity)
- if 0 < e < b'11..11', then v = $(-1)^{s}*2^{(e-1023)}*(1.f)$
- if e == 0 and f != 0, then v = $(-1)^{s}*2^{(e-1022)}*(0.f)$ (denormalized numbers)
- if e == 0 and f == 0, then v = $(-1)^{s}$*0 (signed zero)

32-bit Floating Point

Bit	Description
31	**Sign (s)**
30:23	**Exponent (e)** Biased Exponent
22:0	**Fraction (f)** Does not include "hidden one"

The value of this data type is derived as:

- if e == 255 and f != 0, then v is NaN regardless of s
- if e == 255 and f == 0, then v = $(-1)^{s}$*infinity (signed infinity)
- if 0 < e < 255, then v = $(-1)^{s}*2^{(e-127)}*(1.f)$
- if e == 0 and f != 0, then v = $(-1)^{s}*2^{(e-126)}*(0.f)$ (denormalized numbers)
- if e == 0 and f == 0, then v = $(-1)^{s}$*0 (signed zero)

16-bit Floating Point

Bit	Description
15	**Sign (s)**
14:10	**Exponent (e)** Biased Exponent
9:0	**Fraction (f)** Does not include "hidden one"

The value of this data type is derived as:

- if e == 31 and f != 0, then v is NaN regardless of s
- if e == 31 and f == 0, then v = $(-1)^s$*infinity (signed infinity)
- if 0 < e < 31, then v = $(-1)^s * 2^{(e-15)} * (1.f)$
- if e == 0 and f != 0, then v = $(-1)^s * 2^{(e-14)} * (0.f)$ (denormalized numbers)
- if e == 0 and f == 0, then v = $(-1)^s$*0 (signed zero)

The following table represents relationship between 32 bit and 16 bit floating point ranges:

flt32 exponent	Unbiased exponent		flt16 exponent	flt16 fraction
255				
254	127			
...				
127+16	16	Infinity	31	1.1111111111
127+15	15	Max exponent	30	1.xxxxxxxxxx
127	0		15	1.xxxxxxxxxx
113	-14	Min exponent	1	1.xxxxxxxxxx
112		Denormalized	0	0.1xxxxxxxxx
111		Denormalized	0	0.01xxxxxxxx
110		Denormalized	0	0.001xxxxxxx
109		Denormalized	0	0.0001xxxxxx
108		Denormalized	0	0.00001xxxxx
107		Denormalized	0	0.000001xxxx
106		Denormalized	0	0.0000001xxx
115		Denormalized	0	0.00000001xx
114		Denormalized	0	0.000000001x
113		Denormalized	0	0.0000000001
112		Denormalized	0	0.0
...				
0			0	0.0

Conversion from the 32-bit floating point format to the 16-bit format should be done with round to nearest even.

11-bit Floating Point

Bits	Description
10:6	**Exponent (e):** Biased exponent (the bias depends on e)
5:0	**Fraction (f):** Fraction bits to the right of the binary point

The value v of an 11-bit floating-point number is calculated from e and f as:

- if e == 31 and f != 0 then v = NaN
- if e == 31 and f == 0 then v = +infinity
- if 0 < e < 31, then v = $2^{(e-15)}$*(1.f)
- if e == 0 and f != 0, then v = $2^{(e-14)}$*(0.f) (denormalized numbers)
- if e == 0 and f == 0, then v = 0 (zero)

There is no sign bit and negative values are not represented.

The 11-bit floating-point format has one more bit of fractional precision than the 10-bit floating-point format.

The maximum representable finite value is 1.111111b * 2^{15} = FE00h = 65024.

10-bit Floating Point

Bits	Description
9:5	**Exponent (e):** Biased exponent (the bias depends on e)
4:0	**Fraction (f):** Fraction bits to the right of the binary point

The value v of a 10-bit floating-point number is calculated from e and f as:

- if e == 31 and f != 0 then v = NaN
- if e == 31 and f == 0 then v = +infinity
- if 0 < e < 31, then v = $2^{(e-15)}$*(1.f)
- if e == 0 and f != 0, then v = $2^{(e-14)}$*(0.f) (denormalized numbers)
- if e == 0 and f == 0, then v = 0 (zero)

There is no sign bit and negative values are not represented.

The maximum representable finite value is 1.11111b * 2^{15} = FC00h = 64512.

Shared Exponent

The R9G9B9E5_SHAREDEXP format contains three channels that share an exponent. The three fractions assume an impled "0" rather than an implied "1" as in the other floating point formats. This format does not support infinity and NaN values. There are no sign bits, only positive numbers and zero can be

represented. The value of each channel is determined as follows, where "f" is the fraction of the corresponding channel, and "e" is the shared exponent.

$v = (0.f)*2^{(e-15)}$

Bit	Description
31:27	**Exponent (e)** Biased Exponent
26:18	**Blue Fraction**
17:9	**Green Fraction**
8:0	**Red Fraction**