Permedia3®

Reference Guide

PROPRIETARY AND CONFIDENTIAL INFORMATION

3Dlabs®

3Dlabs®

Permedia3®

Reference Guide™

PROPRIETARY AND CONFIDENTIAL INFORMATION

Issue 8

Proprietary Notice

The material in this document is the intellectual property of **3D***labs*. It is provided solely for information. You may not reproduce this document in whole or in part by any means. While every care has been taken in the preparation of this document, **3D***labs* accepts no liability for any consequences of its use. Our products are under continual improvement and we reserve the right to change their specification without notice. **3D***labs* may not produce printed versions of each issue of this document. The latest version will be available from the **3D***labs* web site.

3D*labs* products and technology are protected by a number of worldwide patents. Unlicensed use of any information contained herein may infringe one or more of these patents and may violate the appropriate patent laws and conventions.

3D*labs* is the worldwide trading name of **3D***labs* Inc. Ltd.

3D*labs*, GLINT and PERMEDIA are registered trademarks of **3D***labs* Inc. Ltd.

Microsoft, Windows and Direct3D are either registered trademarks or trademarks of Microsoft Corp. in the United States and/or other countries. OpenGL is a registered trademark of Silicon Graphics, Inc. All other trademarks are acknowledged and recognized.

Email: info@3dlabs.com
Web: http://www.3dlabs.com

3D*labs* Ltd.
Meadlake Place
Thorpe Lea Road, Egham
Surrey, TW20 8HE
United Kingdom
Tel: +44 (0) 1784 470555
Fax: +44 (0) 1784 470699

3D*labs* K.K
Shiroyama JT Mori Bldg 16F
40301 Toranomon
Minato-ku, Tokyo, 105,
Japan
Tel: +81-3-5403-4653
Fax: +91-3-5403-4646

3D*labs* Inc.
480 Potrero Avenue
Sunnyvale, CA 94086,
United States
Tel: (408) 530-4700
Fax: (408) 530-4701

Change History

Document	Issue	Date	Change
157.2.0	1	19 October 1998	Creation
157.2.0	2	22 November 98	Engineering updates throughout, format corrections
157.2.0	3	10 March 1999	Engineering updates throughout, format corrections
157.2.0	4	10 April 1999	Engineering updates, remove chap. 11 Electrical.
157.2.0	5	10 June 1999	Minor updates, reinstate chap. 12 Electrical, add FBData, FBSourceData, TextureCacheReplacementMode
157.2.0	6	15 September 1999	PCIPllStatus Offset and PostScale reset value corrected, typographical errors, display resolutions
157.2.0	7	02 February 2000	Download Address, download data replaced; corrected FBSourceReadBufferOffset
157.2.0	8	15 April 2000	Added unusued terminations to Pinlist, power-up sequence to Reset.

Table of Contents

List of Figures

List of Tables

1

Functional Overview

1.1 Introduction

PERMEDIA 3 is a high performance PCI/AGP graphics processor that balances high quality 3D polygon and textured graphics acceleration, windows acceleration and state-of-the-art MPEG1/MPEG2 playback with a fast integrated SVGA core, integrated RAMDAC and video ports.

PERMEDIA 3 offers significant advances over earlier members of the PERMEDIA product family in both raw performance and functionality. Specific enhancements include:

Table 1-1 PERMEDIA 3 Enhancement Summary

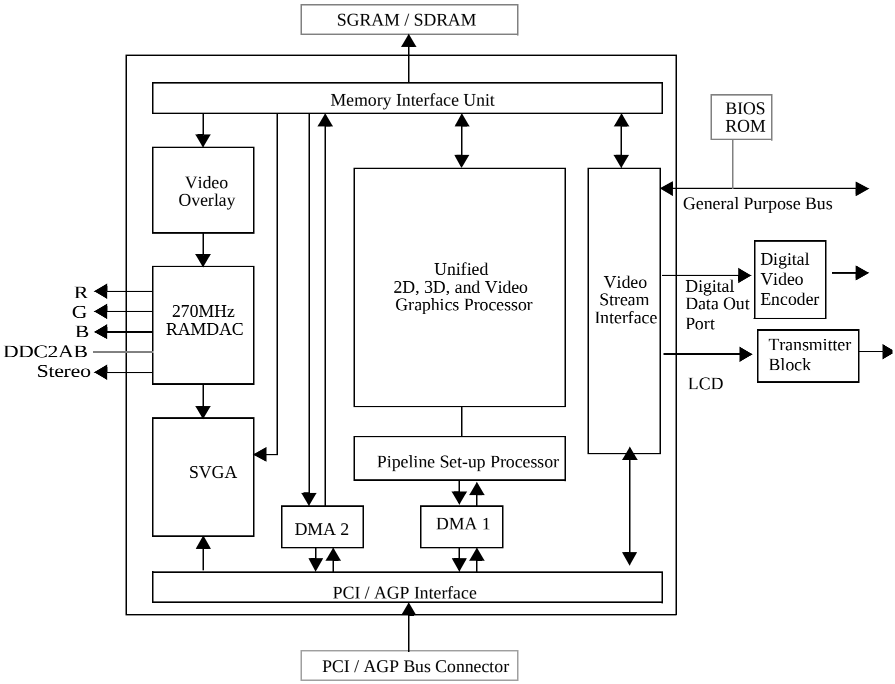
Memory Interface and Core	
Memory bus width (bits)	128
Core clock speed (MHz) - Provisional	125
Max. memory (MB)	32
AGP	2x
RAMDAC speed (MHz)	270
3D	
Max. Z-buffer depth (bits)	32
Non-Linear 16 or 24-bit Z-buffer (Direct3D and OpenGL)	✔
W-Buffer Emulation with Non-Linear Z-buffer (Direct3D)	✔

3D (cont'd)	
Texture read units	2
Texture compositing units	✓
Single-pass dual texturing	✔
Single-pass bump mapping with surface texture	✔
Hardware texture paging	✔
Mip mapping (single pass)	Tri-linear
Destination Alpha	✔
Supports all OpenGL and Direct3D blend modes	✔
Native support for D3D vertex formats	✔
OpenGL	1.2
Fog table	✔
Full hardware edge anti-aliasing	✔
Video/DVD	
Hardware video overlay	✔
Hardware scaling and filtering	✔
MPEG Motion compensation	✔
Memory to DVD accelerator DMA	✔
Flat panel LCD support	✔
Software	
SoftImage Compliant	✔

1.2 Functional Blocks

The major functional blocks are shown in Figure 1-1.

Figure 1-1 Chip Level Block Diagram



1.2.1 AGP/PCI Interface

The PCI interface conforms to the PCI Local Bus standard Revision 2.1. PERMEDIA 3 is a PCI Local Bus Target, a PCI Local Bus Read Master, and a PCILocal Bus Write Master. It is also an AGP read master with support for pipe lined reads and sideband addressing.

The PCI interface has an input FIFO for passing data to the Graphics Core, and an output FIFO for buffering up data to be read from the Graphics Core. The input FIFO is 256 words deep, the output FIFO is 8 words deep. A DMA controller is provided in the PCI interface to allow PERMEDIA 3 to read data directly into the Graphics Core input FIFO or directly out of the output FIFO.

AGP 2X is Intel's high performance, component level interconnect targeted at 3D display applications, which uses a 66MHz PCI specification as an operation baseline and provides three significant performance extension to the PCI specification.

The specification for PERMEDIA 3's AGP implementation is:

- 133 MHz transfer rate (528 Mbytes/s)
- DMA and execute mode support
- Sideband addressing

Implementing these features enables PERMEDIA 3 to achieve 528 Mbytes per second bandwidth from the host for instructions, textures, video data (limited by the host system throughput).

The add-in slot defined for AGP uses a connector body, which is not compatible with the PCI connector, therefore boards designed for use in an AGP slot are not mechanically interchangeable with PCI boards.

1.2.2 Unified 2D/3D/Video Integrated Graphics Processor

The graphics core in PERMEDIA 3 accelerates the key operations for 3D and 2D applications. For further information on the functionality of the graphics processor (GP), refer to the *PERMEDIA 3 Programmer's Manual* and chapter 5, Graphics Registers, in this manual.

1.2.3 Memory Interface

The local memory is used to store color, depth, stencil, and texture data. For more information on the different data types and usage refer to Chapter 9 - Memory System.

The memory is organized as 1 to 4 blocks (blocks 0-3) of SGRAM or SDRAM. The memory interface is 128 bits wide with control lines for 4 blocks of memories (0-3). Block zero must always be fitted as the SVGA uses this area for local storage. Any other combination of banks may be fitted, but for contiguous memory banks should be added from 1 to 3.

PERMEDIA 3 will make use of special SGRAM features including block fill and write-per-bit masking. SDRAM may be used in place of SGRAM if it is identical to SGRAM except for missing block write and write per bit masks.

1.2.4 SVGA and Display Resolutions

The on-chip SVGA unit is register level compatible with standard VGA devices and requires no software emulation. It natively supports all standard VGA modes and certain VESA VBE extended modes.

The standard VESA VBE extended video modes shown below are supported. Those not supportable by the SVGA unit may be supported using the Graphics Processor. Resolution constraints are driver and memory dependant: 1920x1200 is currently supported, but the limits for a 32MByte framebuffer are for example 2048x1200 at 32bit colors, 32bit Z or 2048x1536 at 32bit colors, 16bit Z. At 16bit color, 16bit Z it should be able to display 2400x2400, but this is untested.

Table 1-2 VESA VBE Graphics Modes

Mode (hex)	Pixels	Colors	Window-ed	Lin-ear	Support-able in SVGA	Support-able in GP
0x100	640x400	256	✓	✓	✓	✓
0x101	640x480	256	✓	✓	✓	✓
0x102	800x600	16	✓	✕	✓	✕
0x103	800x600	256	✓	✓	✕	✓
0x104	1024x768	16	✓	✕	✓	✕
0x105	1024x768	256	✓	✓	✕	✓
0x106	1280x1024	16	✓	✕	✓	✕
0x107	1280x1024	256	✓	✓	✕	✓
0x109	320x200	32K (5:5:5:1)	✓	✓	✕	✓
0x10D	320x200	64K (5:6:5)	✓	✓	✕	✓
0x10F	320x200	16.8M (8:8:8)	✓	✓	✕	✓
0x110	640x480	32K (5:5:5:1)	✓	✓	✕	✓
0x111	640x480	64K (5:6:5)	✓	✓	✕	✓
0x112	640x480	16.8M (8:8:8)	✓	✓	✕	✓
0x113	800x600	32K (5:5:5:1)	✓	✓	✕	✓
0x114	800x600	64K (5:6:5)	✓	✓	✕	✓
0x115	800x600	16.8M (8:8:8)	✓	✓	✕	✓
0x116	1024x768	32K (5:5:5:1)	✓	✓	✕	✓
0x117	1024x768	64K (5:6:5)	✓	✓	✕	✓
0x118	1024x768	16.8M (8:8:8)	✓	✓	✕	✓
0x119	1280x1024	32K (5:5:5:1)	✓	✓	✕	✓
0x11A	1280x1024	64K (5:6:5)	✓	✓	✕	✓
0x11B	1280x1024	16.8M (8:8:8)	✓	✓	✕	✓

The following VESA VBE text modes are supportable in the SVGA:

Table 1-3 VESA VBE Text Modes

Mode (hex)	Characters (col/row)
0x108	80x60
0x109	132x25
0x10A	132x43
0x10B	132x50
0x10C	132x60

PERMEDIA 3 allows VESA bankswitching to be done through the bypass to enable additional VESA mode support.

ModeX is also supported.

1.2.5 RAMDAC

PERMEDIA 3 incorporates a high performance 270MHz RAMDAC.

Its characteristics include a high resolution 270 MHz 128-bit RAMDAC. It supports screen resolutions up to 1600x1200 with refresh rates of 96Hz or 1920x1080 with refresh rates of 90Hz. It supports packed pixel formats, with color depths of 8, 16, and 32 bits per pixel. It has dot-clock phase locked loops (PLLs) and triple 8-bit D/A converters. The RAMDAC contains a 64x64x2 bit cursor array to support a 2, 4, or 16 color hardware cursor with cursor shapes cache.

1.2.6 Video Overlay

The video overlay is used to display incoming video data on screen. the overlay selection is based on a transparent color, the overlay key, which can be any RGB color or alpha value. Optionally, the overlay can be blended with the main image by using a 2-bit blend factor. A filter process supports zooming and shrinking at any rate. It combines four pixels into one by using bilinear filtering to achieve best results. Furthermore the filtered output is optionally converted from YUV to RGB color space format.

1.2.7 DMA1..DMA3

1.2.7.1 DMA1 Controller - System to Graphics Core and Graphics Core to System

- Autonomous Setup-Fetch parallelism
- No Wait State - maximum transfer rate
- Programable Block Size - large DMA buffers
- Separate DMA Controllers for Upload and Download which can run concurrently

1.2.7.2 DMA2 Controller - System to Memory and Memory to System

- Fast texture image uploads and downlaods
- Separate DMA Controllers for Upload and Download which can run concurrently
- DMA controller supports scatter/gather
- Fast software MPEG2 download - fast frame capture

1.2.7.3 DMA3 Controller - System Memory to DVD accelerator

- Compressed video to DVD accelerator chip Input FIFO
- Fetch/draw parallelism
- Burst mode - bursts for programmed I/O
- DMA controller supports scatter/gather

1.2.8 Video Streaming

PERMEDIA 3 supports digital video output. The 24-bit streamed output is designed to work with common PAL/NTSC encoders or flat panel controllers.

1.2.9 ROM support

PERMEDIA 3 supports a Flash ROM. This ROM may store code needed for device-specific initialization and the SVGA BIOS.

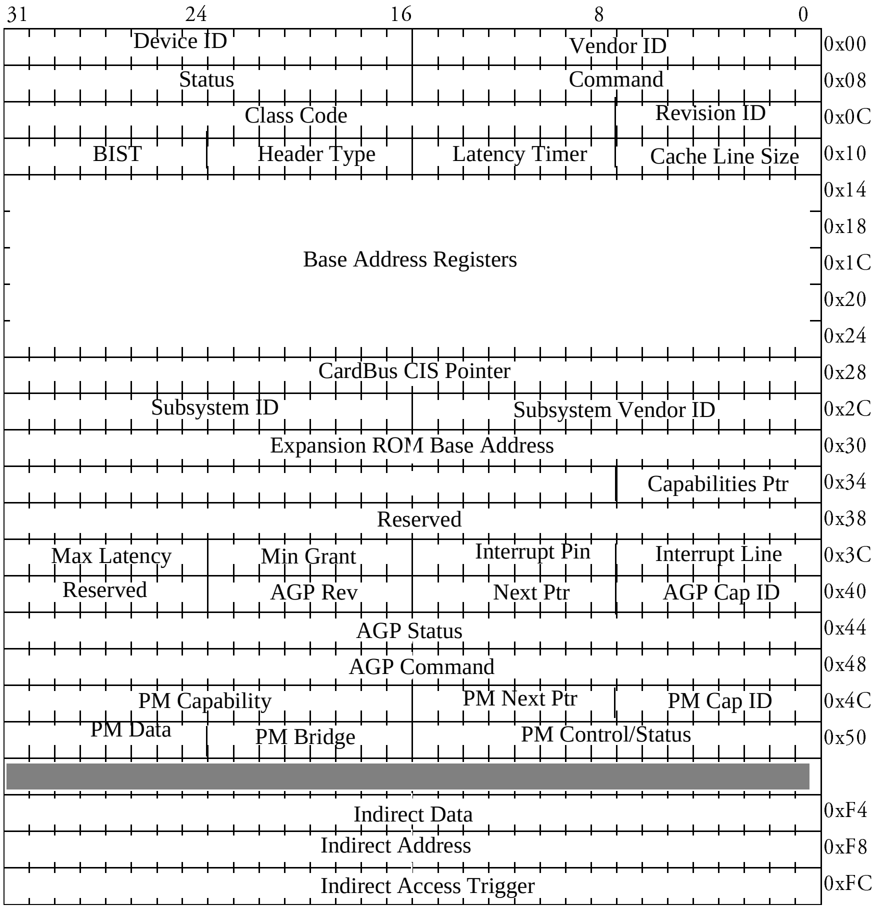
2

Address Maps and Regions

2.1 PCI Configuration Region

The PCI Configuration Region provides information that satisfies the needs of current and anticipated system configuration mechanisms.

Figure 2-1 PCI Configuration Region



2.2 Region Zero Address Map

The PERMEDIA 3 region zero address map is shown in Table 2-1.

Table 2-1 Region Zero Address Map

Address Range	Region Select	Byte Swap/ Write Combined
0000.0000 -> 0000.02FF	Control Status	No
0000.0300 -> 0000.03FF	Bypass Control	No
0000.0400 -> 0000.0FFF	Repeat of the Control and Bypass Decodes	No
0000.1000 -> 0000.1FFF	Memory Control	No
0000.2000 -> 0000.2FFF	GP FIFO Access	No
0000.3000 -> 0000.30FF	Video Control	No
0000.3100 -> 0000.3FFF	Video Overlay Control	No
0000.4000 -> 0000.4FFF	RAMDAC	No
0000.5000 -> 0000.57FF	VS GP	No
0000.5800 -> 0000.5FFF	VSCtl	No
0000.6000 -> 0000.6FFF	VGA Control	No
0000.7000 -> 0000.7FFF	TextureData FIFO	No
0000.8000 -> 0000.FFFF	GP Registers	No
0001.0000 -> 0001.01FF	Control Status	Yes
0001.0300 -> 0001.03FF	Bypass Control	Yes
0001.0400 -> 0001.0FFF	Repeat of the Control and Bypass Decodes	Yes
0001.1000 -> 0001.1FFF	Memory Control	Yes
0001.2000 -> 0001.2FFF	GP FIFO Access	Yes
0001.3000 -> 0001.37FF	Video Control	Yes
0001.3800 -> 0001.3FFF	Video Overlay Control	Yes
0001.4000 -> 0001.4FFF	RAMDAC	Yes
0001.5000 -> 0001.57FF	VS GP	Yes
0001.5800 -> 0001.5FFF	VSCtl	Yes
0001.6000 -> 0001.6FFF	VGA Control	Yes
0001.7000 -> 0001.7FFF	TextureData FIFO	Yes
0001.8000 -> 0001.FFFF	GP Registers	Yes

2.3 PCI Address Regions

PERMEDIA 3 has seven base address regions, as shown in Table 2-2.

Table 2-2 PERMEDIA 3 PCI Address Regions

Region	Address Space	Bytes	Description	Comments
Config	Configuration	256	PCI Configuration	PCI special
Zero	Memory	128 K	Control Registers	Relocatable
One	Memory	64M	Memory Aperture One	Relocatable
Two	Memory	64M	Memory Aperture Two	Relocatable
Three	I/O	16	Indirect Access I/O	Optional and Relocatable
ROM	Memory	64 K	Expansion ROM	Relocatable
VGA	Memory & I/O	—	VGA Access	Optional and Fixed

3

Video Unit and RAMDAC

The video unit and RAMDAC should be configured to display the framebuffer data with the format, resolution, and refresh frequency required.

3.1 Display Timing Values

Table 3-1 Timing Values for 640x480 16 BPP 75Hz

Parameter	Hex	Decimal
HTotal	0x065	101
HsStart	0x003	3
HsEnd	0x00B	11
HbEnd	0x016	22
HgEnd	0x016	22
VTotal	0x1F5	501
VsStart	0x000	0
VsEnd	0x003	3
VbEnd	0x016	22
ScreenStride	0x050	80
ScreenBase	0x000	0
VideoControl	0x029	41

Table 3-2 Timing Values for 800x600 32 BPP 75Hz

Parameter	Hex	Decima l
HTotal	0x103	259
HsStart	0x00A	10
HsEnd	0x01E	30
HbEnd	0x03C	60

Parameter	Hex	Decima l
HgEnd	0x03C	60
VTotal	0x272	626
VsStart	0x000	0
VsEnd	0x003	3
VbEnd	0x01B	27
ScreenStride	0x0C8	200
ScreenBase	0x000	0
VideoControl	0x029	41

4

Hardware Registers

This chapter lists PERMEDIA 3 hardware registers by region and functional offset group. Within each group, the registers are listed alphanumerically. Exceptionally, graphics core "software" registers (offset 8000-9FFF) are shown in chapter 5. Global cross-reference listings in alphanumeric and offset order are available in chapter 6.

Register details have the following format information:

Name	The register's name.
Type	The region in which the register functions.
Offset	The offset of this register from the base address of the region.
Format	Can be bitfield or integer.
Bit	Bit Name
Read	Indicates whether the register bit can be read from. A ✓ mark indicates the register can be read from, a ✕ indicates the register bit is not readable.
Write	Indicates whether the register bit can be written to. A ✓ mark indicates the register can be written to, a ✕ indicates the register bit is not writable.
Reset	The value of the register following hardware reset.
Description	In the register descriptions:
Reserved	Indicates bits that may be used in future members of the PERMEDIA family. To ensure upwards compatibility, any software should not assume a value for these bits when read, and should always write them as zeros.
Not Used / Unused	Indicates bits that are adjacent to numeric fields. These may be used in future members of the PERMEDIA family, but only to extend the dynamic range of these fields. The data returned from a read of these bits is undefined. When a Not Used field resides in the most significant position, a good convention to follow is to sign extend the numeric value, rather than masking the field to zero before writing the register. This will ensure compatibility if the dynamic range is increased in future members of the PERMEDIA family.

For enumeration fields that do not specify the full range of possible values, only the specified values should be used. An example of an enumeration field is the comparison field in the DepthMode register. Future members of the PERMEDIA family may define a meaning for the unused values.

4.1 PCI Configuration Region (0x00-0x30)

CFGAGPCommand

Name	**Type**	**Offset**	**Format**
CFGAGPCommand	Config	0x48	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description	
0..2	DataRate	✓	✓	0	0 = AGP disabled	1 = 1X transfer rate
					2 = 2X transfer rate	4 = 4X transfer rate
					Setting this field to any other value will disable AGP mastering.	
3	Reserved	✓	×	0		
4	FWEnable	✓	✓	0	0 = Fast Write disabled	1 = Fast Write enabled
5	4GEnable	✓	✓	0	0 = 4G Addressing disabled	1 = 4G Addressing enabled
6..7	Reserved	✓	✓	0		
8	AGPEnable	✓	✓	0	0 = AGP Mastering disabled	1 = AGP Mastering enabled
9	SBAEnable	✓	✓	0	0 = sideband addressing disabled	1 = sideband addressing enabled
10..23	Reserved	✓	×	0		
24..31	RQDepth	✓	✓	0	Maximum number of AGP requests that can be queued. The RQDepth set in this field should never exceed the value in the CFGAGPStatus register. The maximum RQDepth used internally is the lower of these two RQDepth fields in case this field has been programmed incorrectly.	

Notes: This register controls the operation of the AGP interface.

- If AGP Capable is not set, writes to this register will be discarded.
- If SBACapable is not set and SBAEnable is set, AGP accesses will be disabled.
- AGP Capable is a term used to express the logical OR of AGP1X Capable with AGP2X Capable with AGP4X Capable.

CFGACGRev

Name	**Type**	**Offset**	**Format**
CFGACGRev	Configuration	0x042	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0..15					See CFGCapID and CFGNextPtr
16..19	Minor Rev	✓	✗	0	Configured by AGP Capbable
					0 when AGP Capable = 0 or 1
20..23	Major Rev	✓	✗	See Desc.	Configured by AGP Capable • 0 when AGP Capable = 0 • 0x2 when AGP Capable = 1
24..31	Reserved	✓	✗	0	

Notes: This register reports the revision of the AGP specification to which the device conforms. AGP Capable is a term used to express the logical OR of AGP1XCapable with AGP2XCapable with AGP4XCapable.

CFGAGPStatus

Name	**Type**	**Offset**	**Format**
CFGAGPStatus	Configuration	0x044	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0..2	Rate	✓	✗	see Desc.	Configured by AGP 1X Capable, Configured by AGP 2X Capable, Configured by AGP 4X Capable 0 = Configured by AGP 1X Capable 1 = Configured by AGP 2X Capable 2 = Configured by AGP 4X Capable
3	Reserved	✓	✗	0	
4	FW	✓	✓	0	
5	4G	✓	✓	0	
9	SBA	✓	✗	see Desc.	Configured by AGP Capable Side Band Addressing 0 when AGP Capable = 0 or SBACapable = 0 1 when AGP Capable = 1 and SBACapable = 1
10..23	Reserved	✓	✗	0	
24..31	RQ	✓	✗	see Desc.	Maximum number of AGP requests supported Configured by AGP Capable 0 if AGP Capable = 0 0x1F if AGP Capable = 1, = 32 outstanding requests

Notes: This register describes the AGP capabilities of the device. AGP Capable is a term used to express the logical OR of AGP1XCapable with AGP2XCapable with AGP4XCapable.

CFGBaseAddr0

Name	**Type**	**Offset**	**Format**
CFGBaseAddr0	Configuration	0x10	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	Memory Space Indicator	✓	×	0	0 = Region is in PCI memory space.
1..2	Address Type	✓	×	0	0 = Memory Space, not prefetchable, in 32 bit address space
3	Prefetchable	✓	×	0	0 = Region is not prefetchable.
4..16	Size Indication	✓	×	0	0 = Control registers must be mapped into 128 Kbytes.
17..31	Base Offset	✓	✓	0	Loaded at boot time to set offset of the control register space (region 0)

Notes: Base Address 0 Register contains the PERMEDIA 3 control space offset. The control registers are in memory space. They are prefetchable and can be located anywhere in 32 bit address space.

CFGBaseAddr1

Name	**Type**	**Offset**	**Format**
CFGBaseAddr1	Configuration	0x14	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	Memory Space Indicator	✓	×	0	0 Region is in PCI memory space.
1..2	Address Type	✓	×	0	0 Locate anywhere in 32 bit address space
3	Prefetchable	✓	×	0	0 = Region is not prefetchable if PrefetchEnable =0. 1= Region is prefetchable if PrefetchEnable = 1.
4..25	Size Indication	✓	×	0	0 = Region size of 64Mbytes.
26..31	Base Offset	✓	✓	0	Loaded at boot time to set offset of the memory space for aperture one.

Notes: The Base Address 1 Register contains the PERMEDIA 3 aperture one memory offset. It is prefetchable and can be located anywhere in 32 bit address space

CFGBaseAddr2

Name	**Type**	**Offset**	**Format**
CFGBaseAddr2	Configuration	0x18	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	Memory Space Indicator	✓	✗	0	0 = Region is in PCI memory space.
1..2	Address Type	✓	✗	0	0 = Locate anywhere in 32 bit address space
3	Prefetchable	✓	✗	0	0 = Region is not prefetchable if PrefetchEnable =0. 1= Region is prefetchable if PrefetchEnable = 1.
4..22	Size Indication	✓	✗	0	0 = Region size of 64Mbytes.
26..31	Base Offset	✓	✓	0	Loaded at boot time to set offset of the memory space for aperture two.

Notes:
- The Base Address 2 Register contains the PERMEDIA 3 aperture 2 memory offset. It is prefetchable and can be located anywhere in 32 bit address space
- The Base Address 3 Register contains the base address of the PERMEDIA 3 Indirect IO aperture, and defines the size and type of this region.

CFGBIST

Name	**Type**	**Offset**	**Format**
CFGBIST	Configuration	0x0F	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..23					See CFGLatTimer and CFGCacheLine
24..31	BIST	✓	✗	0	0 = BIST unsupported by PERMEDIA 3 over the PCI interface

Notes: Optional register used for control and status of Built-In Self Test (BIST).

CFGCacheLine

Name	Type	Offset	Format
CFGCacheLine	Configuration	0x0C	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..15	Cache Line Size	✓	×	0	0= Cache line size unsupported
8..31					See CFGBist, CFGHeaderType, and CFGLatTimer

Notes: This register specifies the cache line size in units of 32 bit words. It is only implemented for PCI bus masters that use the “memory write and invalidate” command. PERMEDIA 3 does not use this command.

CFGCapID

Name	Type	Offset	Format
CFGCapID	Configuration	0x040	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..7	Capability ID	✓	×	see desc.	Configured by AGP Capable
					0 when AGP Capable = 0 2 when AGP Capable = 1
8..23					See CFGNextPtr, CFGAGPRev and Reserved
24..31	Reserved	×	×	0	

Notes: This register specifies that the device has AGP capability. AGP Capable is a term used to express the logical OR of AGP1XCapable with AGP2XCapable with AGP4XCapable

CFGCapPtr

Name	**Type**	**Offset**	**Format**
CFGCapPtr	Configuration	0x34	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..7	Capability Ptr	✓	×	0x4C	Pointer to Power Management capability, address 0x4C.
8..31	Reserved	×	×	0	

Notes: This register is an eight bit register used to provide an offset into the configuration space for the first item in a capabilities list. It is used to point to the Power Management Capability that commences at offset 0x48

CFGCardBus

Name	**Type**	**Offset**	**Format**
CFGCardBus	Configuration	0x28	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..31	CardBus CIS Pointer	×	×	0	0 = Not implemented

Notes:

CFGClassCode

Name	Type	Offset	Format
CFGClassCode	Configuration	0x09	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0..7					See CFGRevisionId
8..15	DeviceClass	✓	✗	from Configuration data	see table below
16..23	SubClass	✓	✗	from Configuration data	see table below
24..31	BaseClass	✓	✗	from Configuration data	see table below

Notes: This device is used to identify the generic function of the PERMEDIA 3 device. This is determined by setting the BaseClassZero and FixedVGAAddressing pins. A more detailed description of the generic function types can be found in Appendix D of the PCI Specification (revisions 2.1 or 2.2).

Configuration Pins					
BaseClass Zero (Config Bit)	**Fixed SVGA Addressing**	**Base Class**	**Sub Class**	**Device Class**	**Generic Function**
0	Disabled	0x03	0x80	0x00	“Other” display controller
0	Enabled	0x03	0x00	0x00	VGA Compatible Controller
1	Disabled	0x00	0x00	0x00	Non-VGA Compatible Controller
1	Enabled	0x00	0x1	0x00	VGA Compatible Device

CFGCommand

Name	**Type**	**Offset**	**Format**
CFGCommand	Configuration	0x04	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description	
0	I/O Space Enable	✓	✗	0	0 = Disable I/O Space Accesses	1 = Enable I/O Space Accesses
					If fixed SVGA addressing is disabled, and indirect I/O region is disabled, this bit will be 0	
1	Memory Space Enable	✓	✓	0	0 = Disable memory Space Accesses	1 = Enable memory Space Accesses
2	Bus Master Enable	✓	✓	0	0 = Disable master access	1 = Enable master access
3	Special Cycle Enable	✓	✗	0	0 = Permedia3 never responds to this special cycle accesses	
4	Memory Write and Invalidate Enable	✓	✗	0	0 = ”Memory Write and Invalidate” is never generated.	
5	SVGA Palette Snoop Enable	✓	✗	0	0 = Treat palette accesses like all other SVGA accesses	1 = Enable SVGA Palette snooping
6	Parity Error Response enable	✓	✗	0	0 = Permedia3 does not support parity error reporting	
7	Address/Data stepping enable	✓	✗	0	0 = Permedia3 does not perform stepping	
8	SERR driver enable	✓	✗	0	0 = Permedia3 does not support parity error reporting	
9	Master Fast Back-to-Back Enable	✓	✗	0	0 = Permedia3 master does not do fast back-to-back accesses	
10..15	Reserved	✓	✗	0		
16..31					See CFGStatus	

Notes: The command register provides control over a device’s ability to generate and respond to PCI cycles. It contains sufficient control bits to fulfill the PERMEDIA 3 PCI functionality. Writing 0 to this register disconnects the device from the PCI for all except configuration accesses

CFGDeviceID

Name	Type	Offset	Format
CFGDeviceID	Configuration *Control register*	0x02	Integer

Bits	Name	Read	Write	Reset	Description
0..15					See CFGVendorID
16..31	DeviceID	✓	✗	0xA	Device identification number: 0x000A = 3Dlabs PERMEDIA 3 device identification number

CFGHeaderType

Name	Type	Offset	Format
CFGHeaderType	Configuration ***Control register***	0x0E	Integer

Bits	Name	Read	Write	Reset	Description
0..15					See CFGLatTimer and CFGCacheLine
16..23	Header Type.	✓	✗	0	PCI Definition: 0 = Single Function Device
24..31					See CFGBist

CFGIndirectAddress

Name	Type	Offset	Format
CFGIndirectAddress	Configuration ***Control register***	0x0F8	Bitfield

Bits	Name	Read	Write	Reset	Description
0..25	Offset	✓	✓	0	Offset within the region.
26..27	Reserved	✓	✗	0	
29..31	Base Address Select	✓	✓	0	0 = Base Address 0 1 = Base Address 1 2 = Base Address 2 3-6 = Reserved 7 = ROM Region

Notes:
1. The Reserved Base Address Select values can be written to or read from the register, but in this case, indirect accesses are treated as if to Base Address 0.
2. Reading the indirect trigger register CFGIndirectTrigger returns the value at the location pointed to by the indirect address register. Indirect data register CFGIndirectData will be written to the location pointed to by the indirect address register CFGIndirectAddress when the indirect trigger register is written.

CFGIndirectData

Name	Type	Offset	Format
CFGIndirectData	Configuration	0x0F4	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..31	Data	✓	✓	0	Data to be written indirectly

Notes: 1. This register is used to access regions 0 to 3 and the ROM region directly through the config space. The region to be accessed and the offset into that region are programmed into the CFGIndirectAddress register. Data written to the CFGIndirectData register will be written to the location pointed to by the CFGIndirectAddress register when the CFGIndirectTrigger register is written.
2. Reading the CFGIndirectTrigger register returns the value at the location pointed to by the CFGIndirectAddress register.

CFGIndirectTrigger

Name	Type	Offset	Format
CFGIndirectTrigger	Configuration	0xFC	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..31	Trigger	✓	✓	0	

Notes: This register is used to trigger indirect accesses as specified by the indirect address and data registers, *CFGIndirectAddress* and *CFGIndirectData*

CFGIntLine

Name	Type	Offset	Format
CFGIntLine	Configuration	0x3C	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..7	Interrupt Line	✓	✓	0	Not read or written by the PERMEDIA 3 device itself.
8..31					See CFGMinGrant, CFGIntPin and CFGMaxLat

Notes: The Interrupt Line register in an 8-bit register used to communicate interrupt line routing information

CFGIntPin

Name	**Type**	**Offset**	**Format**
CFGIntPin	Configuration	0x3D	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..7					See CFGIntLine
8..15	Interrupt Pin	✓	✗	0x1	0x01 = PERMEDIA 3 uses Interrupt pin INTAN
16..31					See CFGMinGrant and CFGMaxLat

Notes: The Interrupt Pin register specifies the interrupt line that PERMEDIA 3 uses.

CFGLatTimer

Name	**Type**	**Offset**	**Format**
CFGLatTimer	Configuration	0x0D	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..7					See CFGCacheLine
8..15	Latency Timer Count	✓	✗	0	Sets the maximum number of PCI clock cycles for master burst accesses.
16..31					See CFGBist and CFGHeaderType

Notes: This register specifies, in PCI bus clocks, the value of the latency timer for this PCI bus master

CFGMaxLat

Name	**Type**	**Offset**	**Format**
CFGMaxLat	Configuration	0x3F	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0-23					See CFGMinGrant, CFGIntPin and CFGIntLine
24-31	Maximum Latency	✓	✗	0xC0	

Notes: This register specifies how often the PCI device needs to gain access to the PCI bus.

CFGMinGrant

Name	**Type**	**Offset**	**Format**
CFGMinGrant	Configuration	0x3E	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0-15					See CFGIntPin and CFGIntLine
16..23	MinimumGrant	✓	✗	0xC0	
24-31					See CFGMaxLat

Notes: This register specifies how long a burst period the PCI device needs.

CFGNextPtr

Name	**Type**	**Offset**	**Format**
CFGNextPtr	Configuration	0x041	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..7					See CFGCapID
8-15	Next Ptr	✓	✗	0	0 = no further capabilities in list
16..23					See CFGAGPRev
24..31	Reserved	✓	✗	0	

Notes: This register points to the next capability data structure. However as there are no more, it is set to zero.

CFGPMC

Name	**Type**	**Offset**	**Format**
CFGPMC	Configuration	0x4E	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0..7					see CFGPMCapID
8..15					see CFGPMNextPtr
16..18	Version	✓	✕	0x1	1 = complies with Revision 1.0 of the PCI Power Management Interface spec.
19	PME clock	✓	✕	0	0 = PME# is not supported in any state
20	Aux Power source	✓	✕	0	0 = PME# is not supported in D3(cold)
21	DSI	✓	✕	1	1 = PERMEDIA 3 requires special initialization following transition to the D0 uninitialized state
22..24	Reserved	✓	✕	0	
25	D1_Support	✓	✕	0x1	1 = D1 power level is supported
26	D2_Support	✓	✕	0	0 = D2 power level is not supported
27..31	PME_Support	✓	✕	0	0 = PME# signal is not asserted in any power state

Notes:

CFGPMCapID

Name	**Type**	**Offset**	**Format**
CFGPMCapID	Configuration	0x4C	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0..7	Power Management Capability ID	✓	✕	0x1	0x01 = Power Management Capability
8..15					See CFGPMNextPtr
16..31					See CFGPMC

Notes: This register specifies that the device has Power Management capability

CFGPMCS

Name	Type	Offset	Format
CFGPMCS	Configuration	0x50	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0..1	PowerState	✓	✓	0	Valid values are 0,1 and 3. If 2 is written to the register, the write is discarded (D2 is not supported) 0 = D0 1 = D1 (This drives the "Low Power" bit internally) 3 = D3(hot)
2..7	Reserved	✓	×	0	
8	PME_EN	✓	×	0	0 = PME# signal is not asserted in D3(cold)
9..12	Data_Select	✓	×	0	0 = Data register not supported
13..14	Data_scale	✓	×	0	0 = Data register not supported
15	PME_Status	✓	×	0	0 = PME# signal is not asserted in D3(cold)
8..15					See CFGPMCSR_BSE
16..31					See CFGPMData

Notes:

CFGPMCSR_BSE

Name	Type	Offset	Format
CFGPMCSR_BSE	Configuration	0x52	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..15					See CFGPCMS
16..23	Power Management Bridge support	✓	×	0	0 = PERMEDIA 3 is not a bridge.
24..31					See CFGPMData

Notes: This register specifies the Power Management PCI-PCI bridge support

CFGPMData

Name	Type	Offset	Format
CFGPMData	Configuration	0x53	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..15					See CFGPCMS
16..23					See CFGPMSR_BSE
24..31	PMData	✓	✗	0	0 = This capapbility is not supported.

Notes: This register is the optional Power Management Data register

CFGPMNextPtr

Name	Type	Offset	Format
CFGPMNextPtr	Configuration	0x4D	
	Control register		

Bits	Name	Read	Write	Reset	Description
0..7					See CFGPMCapID
8..15	Next Ptr	✓	✗	See Desc.	0 = no further capabilities in list if AGP Capable = 0 0x40 = point to AGP Capability if AGP Capable = 1
16..31					See CFGPMC

Notes: This register specifies the device has next capability item. This register reports the revision of the AGP specification to which the device conforms. AGP Capable is a term used to express the logical OR of AGP1XCapable with AGP2XCapable with AGP4XCapable.

CFGRevisionID

Name	**Type**	**Offset**	**Format**
CFGRevisionID	Configuration	0x08	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..7	RevisionID	✓	✗	0x1	Revision Identification Number
8..31					See CFGClassCode

Notes:

CFGRomAddr

Name	**Type**	**Offset**	**Format**
CFGRomAddr	Configuration	0x30	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	Access Decode Enable	✓	✓	0	0= Expansion ROM accesses disabled
					1= Expansion ROM accesses enabled
1..10	Reserved	✓	✗	0	0 = PCI Reserved register bits
11..15	Size Indication	✓	✗	0	0 = Indicates that Expansion ROM must be mapped into 64Kbytes.
16..31	Base Offset	✓	✓	0	Loaded at boot time to set offset of the expansion ROM.

Notes: The expansion ROM base register is the offset address for the expansion ROM.

CFGStatus

Name	Type	Offset	Format
CFGStatus	Configuration	0x06	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0..15					See CFGCommand
16..19	Reserved	✗	✗	0	
20	Cap_List	✓	✗	0x1	1 = PERMEDIA 3 can accept additional capabilities beyond PCI2.1. These are power management and AGP (if AGP Capable is set in CFGCapID)
21	66MHz Capable	✓	✗	X	0 = Permedia3 is 33MHz capable only / 1 = Permedia3 is 66MHz capable
22	UDF Supported	✓	✗	0	0 = Permedia3 does not support user-definable configurations
23	Fast back-to-back capable	✓	✗	0x1	1 = Permedia3 can accept fast back-to-back PCI transactions
24	Data Parity Error Detected	✓	✗	0	0 = Parity checking not implemented on Permedia3
25..26	DEVSEL Timing	✓	✗	0x1	1 = Permedia3 asserts DEVSEL# at medium speed
27	Signaled Target Abort	✓	✗	0	0 = Permedia3 never signals Target-Abort
28	Received Target Abort	✓	✓	0	This bit is set by the Permedia3 bus master whenever its transaction is terminated with Target-Abort
29	Received Master Abort	✓	✓	0	This bit is set by the Permedia3 bus master whenever its transaction is terminated with Master-Abort
30	Signalled System Error	✓	✗	0	0 = Permedia3 never asserts a system error
31	Detected Parity Error	✓	✗	0	0 = Parity checking is not implemented by Permedia3

Notes: Writes to this register causes bits to be reset, but not set. A bit is reset whenever the register is loaded with the corresponding bit position set to one. AGP Capable is a term used to express the logical OR of AGP1XCapable with AGP2XCapable with AGP4XCapable

CFGSubsystemId

Name	Type	Offset	Format
CFGSubsystemId	Configuration ***Control register***	0x02E	Integer

Bits	Name	Read	Write	Reset	Description
0..15					See CFGSubsystemVendorID
16..31	SubsystemId	✗	✓ once	see text	

Notes: This register is used to identify the add-in board on which the PERMEDIA 3 device resides. It has two possible reset states: the value may be loaded from the ROM byte addresses 0xFFFE and 0xFFFF, or reset to the Device ID and then written to once before it becomes read only. The option is controlled by a configuration register

CFGSubsystemVendorId

Name	Type	Offset	Format
CFGSubsystemVendorId	Configuration *Control register*	0x02C	Integer

Bits	Name	Read	Write	Reset	Description
0..15	SubsystemVendorID	✗	✓ once	see text	
16..31					See CFGSubsystemId

Notes: This register is used to identify the vendor of the add-in board on which the PERMEDIA 3 device resides. It has two possible reset states: The value may be loaded from the ROM byte addresses 0xFFFC and 0xFFFD, or reset to the vendor ID and then written to once before it becomes read-only. The option is controlled by a configuration register

CFGVendorID

Name	Type	Offset	Format
CFGVendorID	Configuration *Control register*	0x00	Integer

Bits	Name	Read	Write	Reset	Description
0..15	Vendor ID	✓	✗	0x3D3D	3Dlabs Company Code
16..31					See CFGDeviceID

Notes: Vendor Identification Number

4.2 Region 0 Control Status (0x0000-0x02FF)

AGPControl

Name	**Type**	**Offset**	**Format**
AGPControl	Control Status *Control register*	0x078	Bitfield

Bits	Name	Read	Write	Reset	Description	
0..2	Reserved	✓	×	0		
3	AGP Long Read Disable	✓	✓	0	0 = AGP Long Read Requests may be generated.	1 = AGP Long Read Requests disabled.
4	Reserved	✓	×	0		
5	AGP Data Fifo throttle	✓	✓	0	0 = RBF# throttle start of data transfer for low priority reads.	1 = Only request data when space is available in AGP data fifo to start receiving the burst (RBF# never asserted)
6	AGP High Priority	✓	✓	0	0 = Use AGP Low Priority reads.	1 = Use AGP High Priority reads
7..31	Reserved	✓	×	0		

Notes: The AGP control register sets up the AGP master.

ApertureOne
ApertureTwo

Name	Type	Offset	Format
ApertureOne	Control Status	0x50	Bitfield
ApertureTwo	Control Status	0x58	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description	
0..7	Reserved	✓	✗	0		
8	VGA Access	✓	✓	0	0 = Address memory controller directly.	1 = Address memory through SVGA subsystem.
9	ROM Access	✓	✓	0	0 = Use this aperture to access memory (SVGA or direct).	1 = Use this aperture to access the Expansion ROM.
10..31	Reserved	✓	✗	0		

Notes: Two memory apertures are provided, each being a PCI region with a fixed size of 64 MBytes. A variety of different access modes are possible - these are now controlled in the Bypass controller registers. The ApertureOne and ApertureTwo registers allow the Apertures to be used to access the SVGA or the ROM instead of the memory controller

ChipConfig

Name	Type	Offset	Format
ChipConfig	Control Status	0x70	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	BaseClassZero	✓	✓	X	0 = Use the correct PCI Base Class Code 1 = Force PCI Base Class Code to be zero
1	VGAEnable	✓	✓	X	0 = Disable internal SVGA subsystem 1 = Enable internal SVGA subsystem
2	VGAFixed	✓	✓	X	0 = Disable SVGA fixed address decoding 1 = Enable SVGA fixed address decoding
3..4	Reserved	✓	✗	X	
5	RetryDisable	✓	✓	X	0 = Enable PCI Retry using "Disconnect-Without-Data" 1 = Disable PCI Retry using "Disconnect-Without-Data"
6	Reserved	✓	✗	X	

7	ShortReset	✓	✓	X	0 = Generate normal "AReset" pulse to rest of the chip 1 = Generate short "AReset" pulse (BusReset+ 64 clocks)
8	SBA Capable	✓	✓	X	0 = AGP sideband Addressing Disable 1 = AGP sideband Addressing Enable
9	AGP 1X Capable	✓	✓	X	0 = Not AGP 1X Capable 1 = AGP 1X Capable
10	AGP 2X Capable	✓	✓	X	0 = Not 2X Capable 1 = 2X Capable
11	AGP 4X Capable	✓	✓	X	0 = Not 4X Capable 1 = 4X Capable
12	SubsystemFromRom	✓	✓	X	0 = Leave subsystem registers with reset values 1 = Load subsystem registers from ROM after reset
13	IndirectIOEnable	✓	✓	X	0 = Base Address 3 disabled - Indirect IO accesses cannot be performed 1 = IndirectIO accesses enabled
14	WC Enable	✓	✓	X	0 = Upper half of region zero is a byte swapped version of lower half 1 = Upper half of region zero is flagged as a Write combined version of the lower half
15	Prefetch Enable	✓	✓	X	0 = Regions 1 and 2 marked as not prefetchable 1 = Regions 1 and 2 marked as prefetchable
16..27	Reserved	✓	×	X	(all bits zero)
28..31	Mask rev	✓	×	See Desc.	Value gives the Mask Revision. The initial revision is 0x0.

Notes: Most of the sampled values from the configuration pins are loaded into the ChipConfig register on the trailing edge of reset. This register can then be read back over the PCI bus, to allow the host to determine how the Permedia 3 chip has been configured, and to modify various fields of the configuration if required.

ControlDMAAddress

Name	Type	Offset	Format
ControlDMAAddress	Control Status *Control register*	0x28	Integer

Bits	Name	Read	Write	Reset	Description
0..31	Control DMA Start Address	✓	✓	0	PCI start address for PCI master read transfer to the graphics processor input fifo.

Notes: When using the GPIn FIFO DMA controller to load the graphics processor, the Control DMA Start Address register should be loaded with the PCI address of the first word in the buffer to be transferred. Writing to the Control DMA Start Address register loads the address into the Control DMA address counter. Once a DMA has been set off, the next Control DMA start address may be loaded. A read of this register returns the last start value loaded even if the DMA is already underway.

ControlDMAControl

Name	Type	Offset	Format
ControlDMAControl	Control Status *Control register*	0x60	Bitfield

Bits	Name	Read	Write	Reset	Description
0	ControlDMA Byte Swap Control	✓	✓	0	This field should only be changed when the ControlDMA controller 0 = Standard. 1 = Byte Swapped is idle.
1	ControlDMA using AGP	✓	✓	0	0 = DMA uses PCI Master 1 = DMA uses AGP Master
2..31	Reserved	✓	×	0	

Notes: The DMA control register sets up the data transfer modes for the DMA controller. Data transfer can be set to byte swapped for big endian hosts.

ControlDMACount

Name	Type	Offset	Format
ControlDMACount	Control Status *Control register*	0x30	Integer

Bits	Name	Read	Write	Reset	Description
0..15	Control DMA Count	✓	✓	0	Number of words to be transferred in the DMA operation. The valid range for this register is 0 to 65535. The register behaviour is undefined if it is written to while non-zero and Mastering is enabled. Mastering is enabled if ControlDMAUseAGP = 0 and PCI Bus Master Enabled or ControlDMAUseAGP = 1 and AGP Master is enabled. See DMAControlRegister.
16..31	Reserved	✓	×	0	

Notes:
1. When using the GPIn FIFO DMA controller to load the graphics processor, the Control DMA Start Address register should be loaded with the PCI address of the first word in the buffer to be transferred. Writing to the Control DMA Start Address register loads the address into the Control DMA address counter. Once a DMA has been set off, the next Control DMA start address may be loaded. A read of this register returns the last start value loaded even if the DMA is already underway.
2. Some bits in this register are set during operation and cleared by writing to the register with those bits set. The bits are DataValid, Start and Stop.

ErrorFlags

Name	Type	Offset	Format
ErrorFlags	Control Status *Control register*	0x38	Bitfield

Bits	Name	Read	Write	Reset	Description
0	Input FIFO Error Flag	✓	✓	0	Flag set on write to full input FIFO. 0 = No error. 1 = Error outstanding.
1	Output FIFO Error Flag	✓	✓	0	Flag set on read from empty output FIFO. 0 = No error. 1 = Error outstanding.
2	Reserved	✓	×	0	
3	Control DMA Error Flag	✓	✓	0	Flag set for direct or register access to input FIFO while DMA is in progress (i.e. when the Control DMACount register is not zero). 0 = No error. 1 = Error outstanding.
4	Video Fifo Underflow Error Flag	✓	✓	0	Flag set when video FIFO underflows 0 = No error 1 = Error outstanding
5	Video Stream B Underflow Error Flag	✓	✓	0	Flag set when video stream B FIFO underflows 0 = No error. 1 = Error outstanding.
6	Video Stream A Overflow Error Flag	✓	✓	0	Flag set when video stream A FIFO Overflows 0 = No error. 1 = Error outstanding.
7	PCI Master Error Flag	✓	✓	0	Flag set when either Master abort or Target abort occurs while PCI Master access in progress. - The CFGStatus register can be read to determine the type of error. 0 = No error. 1 = Error outstanding.
8	GPOutDMA Error Flag	✓	✓	0	Flag set for slave access to output FIFO while DMA is in progress 0 = No error. 1 = Error outstanding.
9	Control DMA Count Overwrite Error Flag	✓	✓	0	Flag set if an attempt is made to write the Control DMACount register when it is not zero. 0 = No error. 1 = Error outstanding.
10	GPOutDMA Feedback Error Flag	✓	✓	0	Flag set if a feedback error occurs. 0 = No error. 1 = Error outstanding.
11	VSA Invalid Interlace Error Flag	✓	✓	0	Flag set if invalid interlace is detected on video stream A. 0 = No error. 1 = Error outstanding.
12	VSB Invalid Interlace Error Flag	✓	✓	0	Flag set if invalid interlace is detected on video stream B. 0 = No error. 1 = Error outstanding.

13	HostIn DMA Error Flag	✓	✓	0	Flag set if HostIN DMA error occurs 0 = No error 1 = Error Outstanding
14..31	Reserved	✓	×	0	

Notes: The Error Flags register shows which errors are outstanding in PERMEDIA3 . Flag bits are reset by writing to this register with the corresponding bit set to a one. Flags at positions where the bits are set to zero will be unaffected by the write.

FIFODiscon

Name	Type	Offset	Format
FIFODiscon	Control Status *Control register*	0x68	Bitfield

Bits	Name	Read	Write	Reset	Description
0	Input FIFO Disconnect Enable	✓	✓	0	0 = Disabled 1 = Enabled
1	Output FIFO Disconnect Enable	✓	✓	0	0 = Disabled 1 = Enabled
2	Texture FIFO Disconnect Enable	✓	✓	0	0 = Disabled 1 = Enabled
3..31	Reserved	✓	×	0	

Notes: The FIFODiscon register enables the input and output FIFO disconnect signals, which drive two physical pins on the PERMEDIA 3. Disconnects are disabled at reset. It also allows protocol disconnects to be enabled for the Texture FIFO.

GPOutDMAAddress

Name	Type	Offset	Format
GPOutDMAAddress	Control Status *Control register*	0x080	Integer

Bits	Name	Read	Write	Reset	Description
0..31	GPOutDMAA ddress	✓	×	0	Next address to be issued to the DMA Arbiter.

Notes: The *GPOutDMA* Address register can be used to monitor the progress of the GPOutDMA controller. It returns the next address to be issued to the DMA arbiter.

HostTextureAddress

Name	Type	Offset	Format
HostTextureAddress	Control Status *Control register*	0x0100	Integer

Bits	Name	Read	Write	Reset	Description
0..3	Reserved	✓	✕	0	.
4..31	HostTextureAddress	3	3	X	

Notes: Used in "Slave Download Mode" to supply the address of the first word of a texture

InFIFOSpace

Name	Type	Offset	Format
InFIFOSpace	Control Status *Control register*	0x18	Integer

Bits	Name	Read	Write	Reset	Description
0..31	Input FIFO Space	✓	✕	128	The number of empty words in the input FIFO. This number of words can be updated before checking "InFIFOSpace" again.

Notes: The InFIFOSpace register shows the number of words that can currently be written to the input FIFO. This register can be read at any time. If the DMA controller for the FIFO is in use, the value read is a snapshot of the current FIFO status.

IntEnable

Name	Type	Offset	Format
IntEnable	Control Status *Control register*	0x08	Bitfield

Bits	Name	Read	Write	Reset	Description
0	Control DMA Interrupt Enable	✓	✓	0	0 = Disable interrupt. 1 = Enable interrupt.
1	Sync Interrupt Enable	✓	✓	0	0 = Disable interrupt. 1 = Enable interrupt

2	Reserved	✓	×	0	
3	Error Interrupt Enable	✓	✓	0	0 = Disable interrupt. 1 = Enable interrupt.
4	Vertical Retrace Interrupt Enable	✓	✓	0	0 = Disable interrupt. 1 = Enable Interrupt .
5	Scanline Interrupt Enable	✓	✓	0	0 = Disable interrupt. 1 = Enable Interrupt
6	Texture DownLoad Interrupt Enable	✓	✓	0	0 = Disable interrupt. 1 = Enable interrupt
7	Bypass DMA Read Interrupt Enable	✓	✓	0	0 = Disable interrupt. 1 = Enable interrupt
8	VSB Interrupt Enable	✓	✓	0	0 = Disable interrupt. 1 = Enable interrupt
9	VSA Interrupt Enable	✓	✓	0	0 = Disable interrupt. 1 = Enable interrupt
10	VS Serial Interrupt Enable	✓	✓	0	0 = Disable interrupt. 1 = Enable interrupt.
11	VidDDC Interrupt Enable	✓	✓	0	0 = Disable interrupt. 1 = Enable interrupt
12	VS External Interrupt Enable	✓	✓	0	0 = Disable interrupt. 1 = Enable interrupt
13	Bypass DMA Write Interrupt Enable	✓	✓	0	0 = Disable interrupt. 1 = Enable interrupt
14	HostIn Command Interrupt Enable	✓	✓	0	0 = Disable interrupt. 1 = Enable interrupt.
15	VS DMA Interrupt enable	✓	✓	0	0 = Disable interrupt 1 = Enable interrupt
16..31	Reserved	✓	×	0	Read Only.

Notes: The IntEnable register selects which internal conditions are permitted to generate a bus interrupt. At reset all interrupt sources are disabled

IntFlags

Name	**Type**	**Offset**	**Format**
IntFlags	Control Status *Control register*	0x10	Bitfield

Bits	Flag Name	Read	Write	Reset	Description
0	Control DMA	✓	✓	0	0 = No interrupt. 1 = Interrupt outstanding.
1	Sync	✓	✓	0	0 = No interrupt. 1 = Interrupt outstanding
2	Reserved	✓	×	0	
3	Error	✓	✓	0	0 = No interrupt. 1 = Interrupt outstanding.
4	Vertical Retrace	✓	✓	0	0 = No interrupt. 1 = Interrupt outstanding.
5	Scanline	✓	✓	0	0 = No interrupt. 1 = Interrupt outstanding
6	Texture Download	✓	✓	0	0 = No interrupt. 1 = Interrupt outstanding
7	Bypass Read DMA	✓	✓	0	0 = No interrupt. 1 = Interrupt outstanding.
8	VSB	✓	✓	0	0 = No interrupt. 1 = Interrupt outstanding.
9	VSA	✓	✓	0	0 = No interrupt. 1 = Interrupt outstanding
10	VS Serial	✓	✓	0	0 = No interrupt. 1 = Interrupt outstanding
11	VidDDC	✓	✓	0	0 = No interrupt. 1 = Interrupt outstanding
12	VS External	✓	✓	0	0 = No interrupt. 1 = Interrupt outstanding.
13	Bypass Write DMA	✓	✓	0	0 = No interrupt. 1 = Interrupt outstanding.
14	HostIn Command DMA	✓	✓	0	0 = No interrupt. 1 = Interrupt outstanding
15	VS DMA	✓	✓	0	0 = No interrupt 1 = Interrupt Outstanding
16..30	Reserved	✓	×	0	
31	VGA Interrupt Line	✓	×	0	0 = No interrupt. 1 = Interrupt asserted.

Notes: The IntFlags register shows which interrupts are outstanding. Flag bits are reset by writing to this register with the corresponding bit set to a one. Flags at positions where the bits are set to zero will be unaffected by the write. (The exception is bit 31, which is read-only and reflects the state of the interrupt line from the VGA. The VGA Interrupt must be enabled and reset by accessing the VGA directly, but is visible in this register for convenience.)

LogicalTexturePage

Name	Type	Offset	Format
LogicalTexturePage	Control Status *Control register*	0x118	Integer

Bits	Name	Read	Write	Reset	Description
0..15	LogicalTexture Page	3	5	X	
16..31	Reserved	3	5	0	

Notes: Used with Slave Download Mode to complete the Texture FIFO protocol..

OutFIFOWords

Name	Type	Offset	Format
OutFIFOWords	Control Status *Control register*	0x0020	Integer

Bits	Name	Read	Write	Reset	Description
0..31	Output FIFO Words	✓	×	0	The number of valid words in the output FIFO. This number of words can be read before checking "OutFIFOWords" again.

Notes: The OutFIFOWords register shows the number of words currently in the output FIFO. This register can be read at any time.

PCIAbortAddress

Name	Type	Offset	Format
PCIAbortAddress	Control Status *Control register*	0x098	Integer

Bits	Name	Read	Write	Reset	Description
0..31	PCIAbortAddr ess	✓	×	0	

Notes: The PCIAbortAddress register contains the first PCI Address issued by the PCI Master to cause an Abort.

PCIAbortStatus

Name	Type	Offset	Format
PCIAbortStatus	Control Status *Control register*	0x090	Bitfield

Bits	Name	Read	Write	Reset	Description
0..6	ReadSource	✓	×	0	The read source in the DMA Arbiter that caused the Abort.
7	ReadStatus	✓	×	0	0 = No read abort 1 = Read abort
8..14	WriteSource	✓	×	0	The Write source in the DMA Arbiter which caused the Abort.
15	WriteStatus	✓	×	0	0 = No Write abort 1 = Write abort.
16..31	Reserved	✓	×	0	

Notes: The PCIAbortStatus register reports whether a PCI Master read or write operation has caused an abort (either a Master Abort or Target Abort.) . The PCIAbortAddress register can be read to determine the first PCI Address issued which caused an abort. The PCIAbortStatus register can be cleared by writing any value to the register.

PCIFeedbackCount

Name	Type	Offset	Format
PCIFeedbackCount	Control Status *Control register*	0x088	Integer

Bits	Name	Read	Write	Reset	Description
0..31	PCI Feedback Count	✓	×	0	Number of words that have been transferred in the DMA operation.

Notes: The PCIFeedbackCount register can be read to monitor the progress of a Feedback DMA. The value returned is the number of double words transferred in the current DMA

PCIPLLStatus

Name	**Type**	**Offset**	**Format**
PCIPLLStatus	Control Status *Control register*	0x00F0	Bitfield

Bits	Name	Read	Write	Reset	Description
0..8	PCIPLLSetup	✓	✓	0x 1327	Provides 9 bits of setup for the deskew PLL.
9..11	PCIPLL PostScale	✓	✓	0x1	Divide by 2
12	PCIPLL Enable	✓	✓	0x1	
13..30	Reserved	✓	✗	0	0
31	Reserved	✗	✗	0	Deskew lock

Notes: The PCIPLLStatus register controls the PCI deskew PLL status bits.

ResetStatus

Name	**Type**	**Offset**	**Format**
ResetStatus	Control Status *Control register*	0x00	Integer

Bits	Name	Read	Write	Reset	Description
0..30	Reserved	✓	✗	0	
31	Software Reset Flag	✓	✓	0	0 = GP is ready for use. 1 = GP is being reset and must not be used

Notes: Writing to the reset status register causes a software reset of the graphics processor (GP). The software reset does not reset the bus interface. The reset takes a number of cycles to complete during which the graphics processor should not be used. A flag in the register shows that the software reset is still in progress.

TexDMAAddress

Name	**Type**	**Offset**	**Format**
TexDMAAddress	Control Status *Control register*	0x120	Integer

Bits	Name	Read	Write	Reset	Description
0..31	TexDMA Address	✓	×	X	

Notes: This register returns the address of the last data returned in response to a texture read operation.

TexFIFOSpace

Name	**Type**	**Offset**	**Format**
TexFIFOSpace	Control Status *Control register*	0x128	Integer

Bits	Name	Read	Write	Reset	Description
0..31	TexFIFOSpace	✓	×	0x10	

Notes: This register returns number of 128-bit spaces in the Texture Data FIFO. space is decremented by 1 after four 32-bit writes to the FIFO region. Software must always write in multiples of four 32-bit words.

TextureDownloadControl

Name	Type	Offset	Format
TextureDownloadControl	Control Status *Control register*	0x108	Bitfield

Bits	Name	Read	Write	Reset	Description
0	Texture Download Enable	✓	✓	X	
1	Texture Download Busy	✓	×	X	
2	Texture MemType	✓	✓	X	0 = PCI, 1 = AGP Download
3..7	TextureGranularity	✓	✓	X	
8..12	TextureThreshold	✓	✓	X	
13	SlaveTextureDownload	✓	✓	X	0 = Use Texture DMA for downloads - Slave Writes to the FIFO are discarded.
					1 = Use Slave writes into the FIFO. (slave Reads of FIFO return zero)
14..31	Reserved	✓	×	0	

Notes:

TextureOperation

Name	Type	Offset	Format
TextureOperation	Control Status *Control register*	0x110	Integer

Bits	Name	Read	Write	Reset	Description
0..8	Length	✓	✕	X	
9..10	Memory Pool	✓	✕	X	
11	Host Virt	✓	✕	X	
12..31	Reserved	✓	✕	X	

Notes: Required in Slave Download Mode to complete the Texture FIFO protocol.

VClkRDacCtl

Name	Type	Offset	Format
VClkRDacCtl	Control Status *Control register*	0x40	Bitfield

Bits	Name	Read	Write	Reset	Description
0	VidCtl(0) pin	✓	✓	0	
1	VidCtl(1) pin	✓	✓	0	

Notes: This 2 bit register is used to select which set of RAMDAC control registers is used to control the DClk PLL.

4.3 Region 0 Bypass Controls (0x0300-0x03FF)

ByAperture1Mode
ByAperture2Mode

Name	Type	Offset	Format
ByAperture1Mode	Bypass Control	0x0300	Bitfield
ByAperture2Mode	Bypass Control	0x0328	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0..1	ByteSwap	✓	✓	0	Controls byte swapping on writing to or reading from local memory.
					0 = ABCD (no swap) 1 = BADC (byte swapped) 2 = CDAB (half word swapped) 3 = DCBA
2	PatchEnable	✓	✓	0	Organizes accesses to local memory to fit 2 dimensional patch. 0 = Off 1 = On
3..4	Format	✓	✓	0	Pixel format. YUV formats are converted from planar 420 to 422 format on writing, and from 422 to planar 420 on reads: 0 = Raw 1 = YUYV 2 = UYVY 3 = Reserved
5..6	PixelSize	✓	✓	0	0 = 8 bits 1 = 16 bits 2 = 32 bits 3 = Reserved
7..8	EffectiveStride	✓	✓	0	Stride used to calculate patched address. Should always be bigger or equal to the real stride of the display” 0 = 1024 1 = 2048 2 = 4096 3 = 8192
9..15	PatchOffsetX	✓	✓	0	Adjusts X position within patch.
16..20	PatchOffsetY	✓	✓	0	Adjusts Y position within patch.
21	Buffer	✓	✓	0	0 = Framebuffer 1 = Localbuffer
22..24	DoubleWrite	✓	✓	0	Do two writes for every one received. Defines the boundary on which the second write occurs. A write to an odd multiple of the segment specified causes a write to the corresponding even segment; a write to an even segment causes a write to the odd segment. 0 = Off 1 = 1 Mbyte 2 = 2 Mbytes 3 = 4 Mbytes 4 = 8 Mbytes 5 = 16 Mbytes 6 = 32 Mbytes 7 = Reserved
25..31	Reserved	✓	✗	0	

Notes:

ByAperture1UStart
ByAperture2UStart

Name	Type	Offset	Format
ByAperture1UStart	Bypass Control	0x0318	Integer
ByAperture2UStart	Bypass Control	0x0340	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..23	UStart	✓	✓	X	Number of 128 bit transfers before interpreting data as U.
24..31	Reserved	✓	×	X	

Notes: Used to control the conversion of planar YUV to packed YUV, this register sets the number of transfers to do before interpreting the data as U.

ByAperture1VStart
ByAperture2VStart

Name	Type	Offset	Format
ByAperture1VStart	Bypass Control	0x0320	Integer
ByAperture2VStart	Bypass Control	0x0348	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..23	VStart	✓	✓	X	Number of 128 bit transfers before interpreting data as V.
24..31	Reserved	✓	×	X	

Notes: Used to control the conversion of planar YUV to packed YUV, this register sets the number of transfers to do before interpreting the data as V.

ByAperture1YStart
ByAperture2YStart

Name	Type	Offset	Format
ByAperture1YStart	Bypass Control	0x0310	Integer
ByAperture2YStart	Bypass Control	0x0338	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..23	YStart	✓	✓	X	Number of 128 bit transfers before interpreting data as Y.
24..31	Reserved	✓	×	X	

Notes: Used to control the conversion of planar YUV to packed YUV, this register sets the number of transfers to do before interpreting the data as Y.

ByAperture1Stride
ByAperture2Stride

Name	Type	Offset	Format
ByAperture1Stride	Bypass Control	0x0308	Integer
ByAperture2Stride	Bypass Control	0x0330	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..11	Stride	✓	✓	X	Number of pixels per line.
12..31	Reserved	✓	×	X	

Notes: Sets the stride of the buffer in local memory. Only used when patching or doing YUV format conversions.

ByDMAReadCommandBase

Name	Type	Offset	Format
ByDMAReadCommandBase	Bypass Control *Control register*	0x0378	Integer

Bits	Name	Read	Write	Reset	Description
0..3	Reserved	✓	✗	X	
4..31	Address	✓	✓	X	Base address of command buffer for DMA transfers from system memory to local memory. Always in system memory. Address is 128 bit aligned.

Notes:

ByDMAReadCommandCount

Name	Type	Offset	Format
ByDMAReadCommand Count	Bypass Control *Control register*	0x0380	Integer

Bits	Name	Read	Write	Reset	Description
0..31	Count	✓	✓	X	Number of command packets to transfer.

Notes:

ByDMAReadMode

Name	Type	Offset	Format
ByDMAReadMode	Bypass Control *Control register*	0x0350	Bitfield

Bits	Name	Read	Write	Reset	Description
0..1	ByteSwap	✓	✓	0	Controls byte swapping on writing to or reading from local memory.
					0 = ABCD (no swap) | 1 = BADC (byte swapped) | 2 = CDAB (half word swapped) | 3 = DCBA
2	PatchEnable	✓	✓	0	Organizes accesses to local memory to fit 2 dimensional patch. 0 = Off 1 = On
3..4	Format	✓	✓	0	Pixel format. YUV formats are converted from planar 420 to 422 format on writing, and from 422 to planar 420 on reads. 0 = Raw 1 = YUYV
5..6	PixelSize	✓	✓	0	0 = 8 bits 1 = 16 bits
7..8	EffectiveStride	✓	✓	0	2 = 4096
9..15	PatchOffsetX	✓	✓	0	Adjusts X position within patch.
16..20	PatchOffsetY	✓	✓	0	Adjusts Y position within patch.
21	Buffer	✓	✓	0	0 = Framebuffer 1 = Localbuffer
22	Active	✓	✓	0	Indicates the status of the DMA. 0 = DMA Idle 1 = DMA Running
23	MemType	✓	✓	0	Type of bus protocol to use for DMA.
					0 = PCI 1 = AGP
24..26	Burst	✓	✓	0	Size of burst defined as log2 of burst size.
27	Align	✓	✓	0	Enables alignment of transfers to 64 byte boundaries. 0 = Off 1 = On
28..31	Reserved	✓	✗	0	

Notes: Controls the operation of the DMA controller reading data from system memory and writing it to local memory.

ByDMAReadStride

Name	Type	Offset	Format
ByDMAReadStride	Bypass Control *Control register*	0x0358	Integer

Bits	Name	Read	Write	Reset	Description
0..11	Stride	✓	✓	X	Number of pixels per line.
12..31	Reserved	✓	×	X	

Notes: Sets the stride of the buffer in local memory. Only used when patching or doing YUV format conversions.

ByDMAReadUStart

Name	Type	Offset	Format
ByDMAReadUStart	Bypass Control *Control register*	0x0368	Integer

Bits	Name	Read	Write	Reset	Description
0..23	UStart	✓	✓	X	Number of 128 bit transfers before interpreting data as U.
24..31	Reserved	✓	×	X	

Notes: Used to control the conversion of planar YUV to packed YUV, this register sets the number of transfers to do before interpreting the data as U.

ByDMAReadVStart

Name	Type	Offset	Format
ByDMAReadVStart	Bypass Control *Control register*	0x0370	Integer

Bits	Name	Read	Write	Reset	Description
0..23	VStart	✓	✓	X	Number of 128 bit transfers before interpreting data as V.
24..31	Reserved	✓	×	X	

Notes: Used to control the conversion of planar YUV to packed YUV, this register sets the number of transfers to do before interpreting the data as V.

ByDMAReadYStart

Name	Type	Offset	Format
ByDMAReadYStart	Bypass Control *Control register*	0x0360	Integer

Bits	Name	Read	Write	Reset	Description
0..23	YStart	✓	✓	X	Number of 128 bit transfers before interpreting data as Y.
24..31	Reserved	✓	×	X	

Notes: Used to control the conversion of planar YUV to packed YUV, this register sets the number of transfers to do before interpreting the data as Y.

ByDMAWriteCommand Base

Name	Type	Offset	Format
ByDMAWriteCommand Base	Bypass Control *Control register*	0x03B0	Integer

Bits	Name	Read	Write	Reset	Description
0..3	Reserved	✓	×	X	
4..31	Address	✓	✓	X	Base address of command buffer for DMA transfers from local memory to system memory. Always in local memory. Address is 128 bit aligned.

Notes:

ByDMAWriteCommandCount

Name	Type	Offset	Format
ByDMAWriteCommand Count	Bypass Control	0x03B8	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..31	Count	✓	✓	X	Number of command packets to transfer.

Notes:

ByDMAWriteMode

Name	Type	Offset	Format
ByDMAWriteMode	Bypass Control *Control register*	0x0388	Bitfield

Bits	Name	Read	Write	Reset	Description
0..1	ByteSwap	✓	✓	0	Controls byte swapping on writing to or reading from local memory. 0 = ABCD (no swap) 1 = BADC (byte swapped) 2 = CDAB (half word swapped) 3 = DCBA
2	PatchEnable	✓	✓	0	Organizes accesses to local memory to fit 2 dimensional patch. 0 = Off 1 = On
3..4	Format	✓	✓	0	Pixel format. YUV formats are converted from planar 420 to 422 format on writing, and from 422 to planar 420 on reads. 0 = Raw 1 = YUYV 2 = UYVY 3 = Reserved
5..6	PixelSize	✓	✓	0	0 = 8 bits 1 = 16 bits 2 = 32 bits 3 = Reserved
7..8	EffectiveStride	✓	✓	0	Stride used to calculate patched address. Should always be bigger or equal to the real stride of the display. 0 = 1024 1 = 2048 2 = 4096 3 = 8192
9..15	PatchOffsetX	✓	✓	0	Adjusts X position within patch.
16..20	PatchOffsetY	✓	✓	0	Adjusts Y position within patch.
21	Buffer	✓	✓	0	0 = Framebuffer 1 = Localbuffer
22	Active	✓	✓	0	Indicates the status of the DMA. 0 = DMA Idle 1 = DMA Running
23	MemType	✓	✓	0	Type of bus protocol to use for DMA. 0 = PCI 1 = AGP
24..26	Burst	✓	✓	0	Size of burst defined as log2 of burst size.
27	Align	✓	✓	0	Enables alignment of transfers to 64 byte boundaries.
28..31	Reserved	✓	×	0	

Notes: Controls the operation of the DMA controller reading data from local memory and writing it to system memory.

ByDMAWriteStride

Name	Type	Offset	Format
ByDMAWriteStride	Bypass Control *Control register*	0x0390	Integer

Bits	Name	Read	Write	Reset	Description
0..11	Stride	✓	✓	X	Number of pixels per line.
12..31	Reserved	✓	×	X	

Notes: Sets the stride of the buffer in local memory. Only used when patching or doing YUV format conversions.

ByDMAWriteUStart

Name	Type	Offset	Format
ByDMAWriteUStart	Bypass Control *Control register*	0x03A0	Integer

Bits	Name	Read	Write	Reset	Description
0..23	UStart	✓	✓	X	Number of 128 bit transfers before interpreting data as U.
24..31	Reserved	✓	×	X	

Notes: Used to control the conversion of planar YUV to packed YUV, this register sets the number of transfers to do before interpreting the data as U.

ByDMAWriteVStart

Name	Type	Offset	Format
ByDMAWriteVStart	Bypass Control *Control register*	0x03A8	Integer

Bits	Name	Read	Write	Reset	Description
0..23	VStart	✓	✓	X	Number of 128 bit transfers before interpreting data as V.
24..31	Reserved	✓	×	X	

Notes: Used to control the conversion of planar YUV to packed YUV, this register sets the number of transfers to do before interpreting the data as V.

ByDMAWriteYStart

Name	Type	Offset	Format
ByDMAWriteYStart	Bypass Control *Control register*	0x0398	Integer

Bits	Name	Read	Write	Reset	Description
0..23	YStart	✓	✓	X	Number of 128 bit transfers before interpreting data as Y.
24..31	Reserved	✓	✕	X	

Notes: Used to control the conversion of planar YUV to packed YUV, this register sets the number of transfers to do before interpreting the data as Y.

4.4 Region 0 Memory Control (0x1000-0x1FFF)

LocalMemCaps

Name	Type	Offset	Format
LocalMemCaps	Memory Control Command register	0x1018	Bitfield

Bits	Name	Read	Write	Reset	Description
0..3	Column Address	✓	✓	0	Address bits to use for column address.
4..7	RowAddress	✓	✓	0	Address bits to use for row address.
8..11	BankAddress	✓	✓	0	Address bits to use for bank address.
12..15	ChipSelect	✓	✓	0	Address bits to use for chip select.
16..19	PageSize	✓	✓	0	Page size (units = full width of memory) 0 = 32 units 1 = 64 units, etc
20..23	RegionSize	✓	✓	0xF	Region size (units = full width of memory) 0 = 32 units 1 = 64 units, etc
24	NoPrecharge Opt	×	✓	0	0 = off 1 = on Note that this bit is not readable
25	SpecialMode Opt	✓	✓	0	0 = off 1 = on
26	TwoColorBloc kFill	✓	✓	0	0 = off 1 = on
27	CombineBanks	✓	✓	0	0 = off 1 = on
28	NoWriteMask	✓	✓	0x1	0 = off 1 = on
29	NoBlockFill	✓	✓	0x1	0 = off 1 = on
30	HalfWidth	✓	✓	0x1	0 = off 1 = on
31	NoLookAhead	✓	✓	0x1	0 = off 1 = on

Notes:
1. The ColumnAddress, RowAddress, BankAddress, and ChipSelect fields select the bits of the absolute physical address that are to be used to define corresponding parameters. Each value follows on from the previous one, so the ChipSelect value starts at ColumnAddress + RowAddress + BankAddress and continues for ChipSelect bits.
2. The PageSize field defines the size of the page, and the RegionSize field defines the size of the region of memory that each of the four page detectors should be assigned to (so that it is set to one quarter of the memory size).

LocalMemControl

Name	Type	Offset	Format
LocalMemControl	Memory Control Command register	0x1028	Bitfield

Bits	Name	Read	Write	Reset	Description
0..2	CASLatency	✓	✓	0x3	0 = 0 clocks 1 = 1 clock 2 = 2 clocks 3 = 3 clocks 4 = 4 clocks 5 = 5 clocks 6 = 6 clocks 7 = 7 clocks
3	Interleave	✓	✓	0	0 = off 1 = on
4..21	Reserved	✓	✕	0	
22..31	Mode	✓	✓	0x030	Mode register value used to configure memory. Bit 22 coresponds to bit 0 of register, bit 31 corresponds to bit 9 of register.

Notes: 1. Values are for delays from the current operation to the next. If the delay is set to zero the next operation can follow the current one in the next CLK cycle.
This generally means that the value loaded into the register is the corresponding data sheet value minus one. For example, the data sheet may specify the block write cycle time to be 2 clocks, so the register value would be one because there has to be a one clock delay between block writes.
2. Bits 22 and 31 of LocalMemControl register correspond respectively to bits 0 and 9 of the mode register in the memory device.

LocalMemPowerDown

Name	Type	Offset	Format
LocalMemPowerDown	Memory Control Command register	0x1038	Bitfield

Bits	Name	Read	Write	Reset	Description
0	Enable	✓	✓	0	0 = Off 1 = On
1..16	Reserved	✓	✕	0	
17..31	Delay	✓	✓	0	Timeout in 32 clock units

Notes: Timeout between reseting memory to low power mode in 32 clock units.

LocalMemRefresh

Name	Type	Offset	Format
LocalMemRefresh	Memory Control Command register	0x1030	Bitfield

Bits	Name	Read	Write	Reset	Description	
0	Enable	✓	✓	1	0 = Off	1 = On
1..7	RefreshDelay	✓	✓	0		
8..31	Reserved	✓	✗	0	Delay in 32 clock units	

Notes: Delay between refresh cycles in 32 clock units.

LocalMemTiming

Name	Type	Offset	Format
LocalMemTiming	Memory Control *Command register*	0x1020	Bitfield

Bits	Name	Read	Write	Reset	Description
0..1	TurnOn	✓	✓	0x3	0 = 0 clocks 2 = 2 clocks 3 = 3 clock 1 = 1 clock
2..3	TurnOff	✓	✓	0x3	0 = 0 clocks 1 = 1 clock 2 = 2 clocks 3 = 3 clock
4..5	RegisterLoad	✓	✓	0x3	0 = 0 clocks 1 = 1 clock 2 = 2 clocks 3 = 3 clock
6..7	BlockWrite	✓	✓	0x3	0 = 0 clocks 1 = 1 clock 2 = 2 clocks 3 = 3 clock
8..10	ActivateToCom mand	✓	✓	0x7	0 = 0 clocks 1 = 1 clock 2 = 2 clocks 3 = 3 clocks 4 = 4 clocks 5 = 5 clocks 6 = 6 clocks 7 = 7 clocks
11..13	PrechargeToAc ti vate	✓	✓	0x7	0 = 0 clocks 1 = 1 clock 2 = 2 clocks 3 = 3 clocks 4 = 4 clocks 5 = 5 clocks 6 = 6 clocks 7 = 7 clocks
14..16	BlockWriteTo Pr echarge	✓	✓	0x7	0 = 0 clocks 1 = 1 clock 2 = 2 clocks 3 = 3 clocks 4 = 4 clocks 5 = 5 clocks 6 = 6 clocks 7 = 7 clocks
17..19	WriteTo Precharg e	✓	✓	0x7	0 = 0 clocks 1 = 1 clock 2 = 2 clocks 3 = 3 clocks 4 = 4 clocks 5 = 5 clocks 6 = 6 clocks 7 = 7 clocks

20..23	ActivateTo Precharge	✓	✓	0xF	0 = 0 clocks 1 = 1 clock 2 = 2 clocks 3 = 3 clocks 4 = 4 clocks 5 = 5 clocks 6 = 6 clocks 7 = 7 clocks 8 = 8 clocks 9 = 9 clocks 10 = 10 clocks 11 = 11 clocks 12 = 12 clocks 13 = 13 clocks 14 = 14 clocks 15 = 15 clocks
24..27	RefreshCycle	✓	✓	0xF	0 = 0 clocks 1 = 1 clock 2 = 2 clocks 3 = 3 clocks 4 = 4 clocks 5 = 5 clocks 6 = 6 clocks 7 = 7 clocks 8 = 8 clocks 9 = 9 clocks 10 = 10 clocks 11 = 11 clocks 12 = 12 clocks 13 = 13 clocks 14 = 14 clocks 15 = 15 clocks
28..31	Reserved	✓	✗	0	

Notes: Values are for delays from the current operation to the next. If the delay is set to zero the next operation can follow the curent one in the next clock cycle. This generally means that the value loaded into the register is the corresponding data sheet value minus one. For example, the data sheet may specify the block write cycle time to be 2 clocks, so the register value would be 1 because there has to be a one clock delay between block writes.

MemBypassWriteMask

Name	**Type**	**Offset**	**Format**
MemBypassWriteMask	Memory Control Command register	0x1008	Integer

Bits	Name	Read	Write	Reset	Description
0..31	Mask	✓	✓	0xFFF FFFF F	Per bit control: 0 = mask write, 1 = allow write

Notes: This register determines the bits that get written to memory by way of the bypass.

MemCounter

Name	**Type**	**Offset**	**Format**
MemCounter	Memory Control Command register	0x1000	Integer

Bits	Name	Read	Write	Reset	Description
0..31	Count	✓	✗	0	

MemScratch

Name	Type	Offset	Format
MemScratch	Memory Control *Command register*	0x1010	Integer

Bits	Name	Read	Write	Reset	Description
0..31		✓	✓		Scratch memory

Notes: Scratch memory

RemoteMemControl

Name	Type	Offset	Format
RemoteMemControl	Memory Control Command register	0x1100	Integer

Bits	Name	Read	Write	Reset	Description	
0	TxReadType	✓	✓	0	0 = PCI	1 = AGP
1..31	Reserved	✓	✗	0		

Notes:

4.5 Region 0 GP FIFO (0x2000-0x2FFF)

No 0x2000 series registers are listed.

4.6 Region 0 Video Control (0x3000-0x3FFF)

DisplayData

Name	**Type**	**Offset**	**Format**
DisplayData	Video Control *Control register*	0x3068	Bitfield

Bits	Name	Read	Write	Reset	Description
0	DataIn	✓	✗	X	0 = Data line is low 1 = Data line is high
1	ClkIn	✓	✗	X	0 = Clock line is low 1 = Clock line is high
2	DataOut	✓	✓	1	0 = Drive data line low 1 = Tri-state data line
3	ClkOut	✓	✓	1	0 = Drive clock line low 1 = Tri-state clock line
4	LatchedData	✓	✗	0	0 = Data latched at 0 1 = Data latched at 1
5	DataValid	✓	✓	0	0 = DataIn not valid 1 = DataIn valid
6	Start	✓	✓	0	0 = Has not passed through start state 1 = Has passed through start state
7	Stop	✓	✓	0	0 = Has not passed through stop state 1 = Has passed through stop state
8	Wait	✓	✓	0	0 = Do not insert wait states 1 = Insert wait states
9	UseMonitorID	✓	✓	0	0 = Use DDC 1 = Use MonitorID
10..11	MonitorIDIn[1..0]	✓	✗	X	0 = Data line is low, clock line is low 1 = Data line is high, clock is high 2 = clock is high, data is low 3 = both high
12	Reserved	✓	✗	0	
13..14	MonitorIDOut [1..0]	✗	✓	0x3	0 = Drive data line low 1 = Tri-state data line
15..31	Reserved	✓	✗	0	

Notes: Some bits in this register are set during operation and cleared by writing to the register with those bits set. The bits are DataValid, Start and Stop

FifoControl

Name	Type	Offset	Format
FifoControl	Video Control *Control register*	0x3078	Bitfield

Bits	Name	Read	Write	Reset	Description
0..4	LowThreshold	✓	✓	0x10	Request data from memory with low priority when there are this many spaces in the fifo.
5..7	Reserved	✓	✗	0	
8..12	High Threshold	✓	✓	0x10	Request data from memory with high priority when there are this many spaces in the fifo.
13..15	Reserved	✓	✗	0	
16	Underflow	✓	✓	0	This bit is set by the by the behavioural code. It is cleared by writing a 1 to this bit. 0 = underflow has not occurred 1 = underflow has occured
17..31	Reserved	✓	✗	0	

HbEnd

Name	Type	Offset	Format
HbEnd	Video Control *Control register*	0x3020	Integer

Bits	Name	Read	Write	Reset	Description
0..10	HbEnd	✓	✓	x	First 128 bit unit out of horizontal blank
11..31	Reserved	✓	✗	0	

HgEnd

Name	Type	Offset	Format
HgEnd	Video Control *Control register*	0x3018	Integer

Bits	Name	Read	Write	Reset	Description
1..10	HgEnd	✓	✓	X	Last 128 bit unit in gate period
11..31	Reserved	✓	✗	0	

Notes:

HsEnd

Name	**Type**	**Offset**	**Format**
HsEnd	Video Control *Control register*	0x3030	Integer

Bits	Name	Read	Write	Reset	Description
0..10	HsEnd	✓	✓	X	First 128 bit unit out of horizontal sync.
11..31	Reserved	✓	✗	0	

Notes:

HsStart

Name	**Type**	**Offset**	**Format**
HsStart	Video Control *Control register*	0x3028	Integer

Bits	Name	Read	Write	Reset	Description
0..10	HsStart	✓	✓	X	First 128 bit unit in horizontal sync.
11..31	Reserved	✓	✗	0	

Notes:

HTotal

Name	**Type**	**Offset**	**Format**
HTotal	Video Control *Control register*	0x3010	Integer

Bits	Name	Read	Write	Reset	Description
0..10	HTotal	✓	✓	X	Last 128 bit unit (including horizontal blank period) on screen
11..31	Reserved	✓	✗	0	

Notes:

InterruptLine

Name	Type	Offset	Format
InterruptLine	Video Control	0x3060	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..10	InterruptLine	✓	✓	X	Generate interrupt at start of this line
11..31	Reserved	✓	✗	0	

Notes:

MiscControl

Name	Type	Offset	Format
MiscControl	Video Control	0x3088	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0..1	StripeMode	✓	✓	0	0 = off 1 = primary
					2 = secondary 3 = reserved
2..3	Reserved	✓	✗	0	
4..6	StripeSize	✓	✓	0	0 = 1 line 1 = 2 lines 2 = 4 lines 3 = 8 lines 4 = 16 lines
7	ByteDouble	✓	✓	0	

ScreenBase

Name	Type	Offset	Format
ScreenBase	Video Control	0x3000	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..20	ScreenBase	✓	✓	X	Base address of screen in 128 bit units.
21..31	Reserved	✗	✗	0	

Notes:

ScreenBaseRight

Name	Type	Offset	Format
ScreenBaseRight	Video Control *Control register*	0x3080	Integer

Bits	Name	Read	Write	Reset	Description
0..20	ScreenBase Right	✓	✓	X	Base address of right screen in 128 bit units.
21..31	Reserved	✗	✗	0	

Notes: **ScreenBaseRight** updates may not take effect unless they are followed by a write to **ScreenBase**. This affects secondary chips in Striped mode only.

ScreenStride

Name	Type	Offset	Format
ScreenStride	Video Control *Control register*	0x3008	Integer

Bits	Name	Read	Write	Reset	Description
0..10	ScreenStride	✓	✓	X	Stride between scanlines in 128 bit units.
11...19	Reserved	✓	✓	X	Mask to 0
20..31	Reserved	✗	✗	0	

Notes:

VbEnd

Name	Type	Offset	Format
VbEnd	Video Control *Control register*	0x3040	Integer

Bits	Name	Read	Write	Reset	Description
0..10	VbEnd	✓	✓	X	First scanline out of vertical blank
11..31	Reserved	✓	✗	0	

Notes:

VerticalLineCount

Name	**Type**	**Offset**	**Format**
VerticalLineCount	Video Control *Control register*	0x3070	Integer

Bits	Name	Read	Write	Reset	Description
0..10	VerticalLineCount	✓	×	X	Current vertical line.
11..31	Reserved	✓	×	0	

Notes:

VideoControl

Name	Type	Offset	Format
VideoControl	Video Control *Control register*	0x3058	Bitfield

Bits	Name	Read	Write	Reset	Description
0	Enable	✓	✓	0	0 = GP video disabled 1 = GP video enabled
1	BlankCtl	✓	✓	0	0 = Active High 1 = Active Low
2	LineDouble	✓	✓	0	0 = Off 1 = On
3..4	HSyncCtl	✓	✓	0	0 = Forced High 1 = Active High
					2 = Forced Low 3 = Active Low
5..6	VSyncCtl	✓	✓	0	0 = Forced High 1 = Active High
					2 = Forced Low 3 = Active Low
7	BypassPending	✓	✗	0	Read only bit set when ScreenBase register is loaded. It is cleared when new value in ScreenBase has been used (i.e. during VBlank)
					0 = ScreenBase register data from bypass used 1 = ScreenBase register data from bypass not used yet.
8	Reserved	✓	✗	0	
9..10	BufferSwap	✓	✓	0	0 = SyncOnFrameBlan k 1 = FreeRunning.
					2 = LimitToFrameRate 3 = Reserved
11	Stereo	✓	✓	0	0= Disabled 1 = Enabled.
12	RightEyeCtl	✓	✓	0	0=Active high 1 = Active low
13	RightFrame	✓	✗	0	0 = Displaying left frame 1 = Displaying right frame
14	VideoExtCtrl	✓	✓	0	0 = low, 1 = high. This bit drives the PADVideo ExternalControl pin directly for use in controlling external devices.
15	Reserved	✗	✗	0	Reserved
16..17	SyncMode	✓	✓	0	0 = Independent 1 = SyncToVSA
					2 = SyncToVSB 3 = Reserved
18	PatchEnable	✓	✓	0	0 = Off 1 = On
19..20	PixelSize	✓	✓	0	0 = 8 bits 1 = 16 bits
					2 = 32 bits 3 = Reserved
21	DisplayDisable	✓	✓	0	0 = Off 1 = On
22..27	PatchOffsetX	✓	✓	0	
28..31	PatchOffsetY	✓	✓	0	

Notes:

VideoOverlayBase0
VideoOverlayBase1
VideoOverlayBase2

Name	Type	Offset	Format
VideoOverlayBase0	Video Overlay Control	0x3120	Bitfield
VideoOverlayBase1	Video Overlay Control	0x3128	Bitfield
VideoOverlayBase2	Video Overlay Control	0x3130	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0..25	Address	✓	✓	X	Pixel address.
26..29	Reserved	✓	✕	0	
30..31	MemoryType	✓	✓	X	0 = Framebuffer 1 = Localbuffer 2 = Reserved 3 = Reserved

Notes:

VideoOverlayFieldOffset

Name	Type	Offset	Format
VideoOverlayFieldOffset	Video Overlay Control	0x3170	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0..3	Reserved	✓	✕	0	
4..27	Offset	✓	✓	X	Scale factor as 12.12 2’s complement fixed point value.
28..31	Reserved	✓	✕	0	

Notes:

VideoOverlayFIFOControl

Name	Type	Offset	Format
VideoOverlayFIFOControl	Video Overlay Control *Control register*	0x3110	Bitfield

Bits	Name	Read	Write	Reset	Description
0..15	Low	✓	✓	0	Low threshold
16..31	High	✓	✓	0xFF	High threshold

Notes:

VideoOverlayHeight

Name	Type	Offset	Format
VideoOverlayHeight	Video Overlay Control *Control register*	0x3148	Integer

Bits	Name	Read	Write	Reset	Description
0..11	Height	✓	✓	X	Height of overlay buffer in lines.
12..31	Reserved	✓	×	0	

Notes:

VideoOverlayIndex

Name	Type	Offset	Format
VideoOverlayIndex	Video Overlay Control *Control register*	0x3118	Bitfield

Bits	Name	Read	Write	Reset	Description
0..1	Index	✓	✓	X	Base address register to use when BufferSync is Manual
2..30	Reserved	✓	×	0	
31	Field	✓	✓	X	0 = Odd 1 = Even

Notes:

VideoOverlayMode

Name	**Type**	**Offset**	**Format**
VideoOverlayMode	Video Overlay Control *Control register*	0x3108	Bitfield

Bits	Name	Read	Write	Reset	Description	
0	Enable	✓	✓	0	0 = Off	1 = On
1..3	BufferSync	✓	✓	0	0 = Manual	1 = VideoStreamA
					2 = VideoStreamB	3..7 = Reserved
4	FieldPolarity	✓	✓	0	0 = Normal	1 = Invert
5..6	PixelSize	✓	✓	0	0 = 8 bits	1 = 16 bits
					2 = 32 bits	3 = Reserved
7..9	ColorFormat	✓	✓	0	6..7 = Reserved	
10..11	YUV	✓	✓	0	0 = RGB 2 = YUV444	1 = YUV422 3 = Reserved
12	ColorOrder	✓	✓	0	0 = BGR	1 = RGB
13	LinearColorExt e nsion	✓	✓	0	0 = Off	1 = On
14..15	Filter	✓	✓	0	0 = Off	1 = Full
					2 = Partial (X with zoom)	3 = Reserved
16..17	DeInterlace	✓	✓	0	0 = Off	1 = Bob
					2..3 = Reserved	
18..19	PatchMode	✓	✓	0	0 = Off 2..3 = Reserved	1 = On
20..22	Flip	✓	✓	0	0 = Video	1 = VideoStreamA
					2 = VideoStreamB	3..7 = Reserved
23	MirrorX	✓	✓	0	0 = Off	1 = On
24	MirrorY	✓	✓	0	0 = Off	1 = On
25..31	Reserved	✓	✗	0		

Notes:

The following table shows the bit positions of each component in each color format.

			Internal Color Channels		
Color Format	**Color Order**	**Name**	**R**	**G**	**B**
0	0	8:8:8:8	8@0	8@8	8@16
1	0	4:4:4:4	4@0	4@4	4@8
2	0	5:5:5:1	5@0	5@5	5@10
3	0	5:6:5	5@0	6@5	5@11

4	0	3:3:2	3@0	3@3	2@6
0	1	8:8:8:8	8@16	8@8	8@0
1	1	4:4:4:4	4@8	4@4	4@0
2	1	5:5:5:1	5@10	5@5	5@0
3	1	5:6:5	5@11	6@5	5@0
4	1	3:3:2	3@5	3@2	2@0
5	1	C18	8@0	8@0	8@0

In YUV422 or YUV444 mode the ColorFormat field is ignored. The following bit positions are used:

			Internal Color Channels		
YUV	**Color Order**	**Name**	**Y**	**U**	**V**
0	0	RGB	-	-	-
1	0	YUV444	8@0	8@8	8@16
2	0	YUV422	8@0	8@8	8@8
3	0	Reserved	-	-	-
0	1	RGB	-	-	-
1	1	YUV444	8@16	8@8	8@0
2	1	YUV422	8@8	8@0	8@0
3	1	Reserved	-	-	-

In YUV422 mode the U and V components share the same bits in alternate pixels; U is always in the lower 16 bits and V in the upper 16 bits.

VideoOverlayOrigin

Name	**Type**	**Offset**	**Format**
VideoOverlayOrigin	Video Overlay Control *Control register*	0x3150	Bitfield

Bits	**Name**	**Read**	**Write**	**Reset**	**Description**
0..11	XOrigin	✓	✓	X	X origin of data to display within source buffer.
12..15	Reserved	✓	✕	0	
16..27	YOrigin	✓	✓	X	Y origin of data to display within source buffer.
28..31	Reserved	✓	✕	0	

Notes:

VideoOverlayShrinkXDelta

Name	Type	Offset	Format
VideoOverlayShrinkXDelta	Video Overlay Control *Control register*	0x3158	Bitfield

Bits	Name	Read	Write	Reset	Description
0..3	Reserved	✓	×	0	
4..27	Delta	✓	✓	X	Scale factor as 12.12 2's complement fixed point value.
28..31	Reserved	✓	×	0	

Notes:

VideoOverlayStatus

Name	Type	Offset	Format
VideoOverlayStatus	Video Overlay Control *Control register*	0x3178	Bitfield

Bits	Name	Read	Write	Reset	Description
0	FIFOUnderflow	✓	✓	0	Set by overlay unit, cleared by writing 1.
1..3	Reserved	×	×	0	
4..28	Reserved	✓	×	X	
29..31	Reserved	✓	×	0	

Notes:

VideoOverlayStride

Name	Type	Offset	Format
VideoOverlayStride	Video Overlay Control *Control register*	0x3138	Integer

Bits	Name	Read	Write	Reset	Description
0..11	Stride	✓	✓	X	Stride of overlay buffer in pixels.
12..31	Reserved	✓	×	0	

Notes:

VideoOverlayUpdate

Name	Type	Offset	Format
VideoOverlayUpdate	Video Overlay Control *Control register*	0x3100	Integer

Bits	Name	Read	Write	Reset	Description
0	Enable	✓	✓	0	Set to 1 to enable update, cleared following update.
1..31	Reserved	✓	✗	0	

Notes:

VideoOverlayWidth

Name	Type	Offset	Format
VideoOverlayWidth	Video Overlay Control *Control register*	0x3140	Integer

Bits	Name	Read	Write	Reset	Description
0..11	Width	✓	✓	X	Width of overlay buffer in pixels.
12..31	Reserved	✓	✗	0	

Notes:

VideoOverlayYDelta

Name	Type	Offset	Format
VideoOverlayYDelta	Video Overlay Control *Control register*	0x3168	Integer

Bits	Name	Read	Write	Reset	Description
0..3	Reserved	✓	✗	0	
4..27	Delta	✓	✓	X	Scale factor as 12.12 2's complement fixed point value.
28..31	Reserved	✓	✗	0	

Notes:

VideoOverlayZoomXDelta

Name	**Type**	**Offset**	**Format**
VideoOverlayZoomXDelta	Video Overlay Control *Control register*	0x3160	Integer

Bits	Name	Read	Write	Reset	Description
0..3	Reserved	✓	×	0	
4..16	Delta	✓	✓	X	Scale factor as 1.12 unsigned
17..31	Reserved	✓	×	0	

Notes:

VsEnd

Name	**Type**	**Offset**	**Format**
VsEnd	Video Control *Control register*	0x3050	Integer

Bits	Name	Read	Write	Reset	Description
10..0	VsEnd	✓	✓	X	First scanline out of vertical sync - 1
31..11	Reserved	✓	×	0	

Notes:

VsStart

Name	**Type**	**Offset**	**Format**
VsStart	Video Control *Control register*	0x3048	Integer

Bits	Name	Read	Write	Reset	Description
0..10	VsStart	✓	✓	X	First scanline in vertical sync – 1.
11..31	Reserved	✓	×	0	

Notes:

VTotal

Name	Type	Offset	Format
VTotal	Video Control *Control register*	0x3038	Integer

Bits	Name	Read	Write	Reset	Description
0..10	VTotal	✓	✓	X	Last scanline on screen, including vertical blank period.
11..31	Reserved	✓	×	0	

Notes:

4.7 Region 0 RAMDAC

Direct and Indirect RAMDAC registers are listed separately.

4.7.1 Direct RAMDAC Registers (0x4000-0x4FFF)

RDIndexControl

Name	Type	Offset	Format
RDIndexControl	RAMDAC Control *Control register*	0x4038	Integer

Bits	Name	Read	Write	Reset	Description	
0	AutoIncrement	✓	✓	0	0 = Disabled	1 = Enabled
1..7	Reserved	✓	✗	0		

Notes: The register is accessed directly by reading or writing to the defined address. It is a byte wide and set on an 8 byte boundary in the PCI address range. When accessed from the SVGA it is set on a byte boundary.

RDIndexedData

Name	Type	Offset	Format
RDIndexedData	RAMDAC Control *Control register*	0x4030	Integer

Bits	Name	Read	Write	Reset	Description
0..7	Data	✓	✓	X	

Notes:
1. A read or write to this register will access the register pointed to by the RDIndex register. Following a read or write to this register, the index will be incremented if AutoIncrement is enabled in RDIndexControl.
2. The register is accessed directly by reading or writing to the defined address. It is a byte wide and set on an 8 byte boundary in the PCI address range. When accessed from the SVGA it is set on a byte boundary

RDIndexHigh

Name	Type	Offset	Format
RDIndexHigh	RAMDAC Control *Control register*	0x4028	Integer

Bits	Name	Read	Write	Reset	Description
0..2	Index	✓	✓	X	
3..7	Reserved	✓	✗	0	

Notes:
1. This register, with RDIndexLow, selects the register that will be accessed when the RDIndexedData register is written or read.
2. The register is accessed directly by reading or writing to the defined address. It is a byte wide and set on an 8 byte boundary in the PCI address range. When accessed from the SVGA it is set on a byte boundary

RDIndexLow

Name	Type	Offset	Format
RDIndexLow	RAMDAC Control *Control register*	0x4020	Integer

Bits	Name	Read	Write	Reset	Description
0..7	Index	✓	✓	X	

Notes:
1. This register, with RDIndexHigh, selects the register that will be accessed when the RDIndexedData register is written or read.
2. The register is accessed directly by reading or writing to the defined address. It is a byte wide and set on an 8 byte boundary in the PCI address range. When accessed from the SVGA it is set on a byte boundary

4.7.2 Indirect RAMDAC Registers (0x200-0xFFF)

RDCheckControl

Name	Type	Offset	Format
RDCheckControl	RAMDAC Control *Control register*	0x018	Bitfield

Bits	Name	Read	Write	Reset	Description
0	Pixel	✓	✓	0	Set to start checksum, cleared when complete. 0 = Disabled 1 = Enabled
1	LUT	✓	✓	0	Set to start checksum, cleared when complete. 0 = Disabled 1 = Enabled
2..7	Reserved	✓	×	0	

Notes:
- This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.
- You can use this register to tell the RAMDAC to sum the R, G and B values for a scan line. Typically, wait for Vblank, enable checksum before or after LUT, wait for RAMDAC to sum first active scanline (after which enable bits are Reset) then read RDCheckLUT* or RDCheckPixel* registers for the corresponding RGB component values.

RDCheckLUTBlue

Name	Type	Offset	Format
RDCheckLUTBlue	RAMDAC Control *Control register*	0x01E	Integer

Bits	Name	Read	Write	Reset	Description
0..7	CheckSum	✓	×	X	Checksum for blue component after look-up table.

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.

RDCheckLUTGreen

Name	Type	Offset	Format
RDCheckLUTGreen	RAMDAC Control *Control register*	0x01D	Integer

Bits	Name	Read	Write	Reset	Description
0..7	CheckSum	✓	×	X	Checksum for green component after look-up table.

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.

RDCheckLUTRed

Name	Type	Offset	Format
RDCheckLUTRed	RAMDAC Control ***Control register***	0x01C	Integer

Bits	Name	Read	Write	Reset	Description
0..7	CheckSum	✓	✕	X	Checksum for red component after look-up table.

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.

RDCheckPixelBlue

Name	Type	Offset	Format
RDCheckPixelBlue	RAMDAC Control *Control register*	0x01B	Integer

Bits	Name	Read	Write	Reset	Description
0..7	CheckSum	✓	✕	X	Checksum for blue component after pixel processing.

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.

RDCheckPixelGreen

Name	Type	Offset	Format
RDCheckPixelGreen	RAMDAC Control *Control register*	0x01A	Integer

Bits	Name	Read	Write	Reset	Description
0..7	CheckSum	✓	✕	X	Checksum for green component after pixel processing.

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.

RDCheckPixelRed

Name	Type	Offset	Format
RDCheckPixelRed	RAMDAC Control *Control register*	0x019	Integer

Bits	Name	Read	Write	Reset	Description
0..7	CheckSum	✓	✗	X	Checksum for red component after pixel processing.

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.

RDColorFormat

Name	Type	Offset	Format
RDColorFormat	RAMDAC Control *Control register*	0x004	Bitfield

Bits	Name	Read	Write	Reset	Description
0..4	ColorFormat	✓	✓	X	See table below
5	RGB	✓	✓	X	Color ordering, see table below.
6	LinearColorExtension	✓	✓	X	0 = Disabled - pad low order bits of components less than 8 bits with zeros. 1 = Enabled - linearly extend low order bits of components less than 8 bits.
7	Reserved	✓	✗	0	

Notes:
1. This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.
2. The table below shows the bit positions for each color format specified. The color format is defined in the form number of bits @ bit position, where the bit position defines the first bit of the component with sucessive bits at incresing bit positions.

			Internal Color Channels			
ColorFormat	**RGB**	**Name**	**R**	**G**	**B**	**O**
0	0	8:8:8:8	8@0	8@8	8@16	8@24
1	0	5:5:5:1Front	5@0	5@5	5@10	1@15
2	0	4:4:4:4	4@0	4@4	4@8	4@12
3	0	Reserved	8@0	8@8	8@16	8@24
4	0	Reserved	8@0	8@8	8@16	8@24
5	0	3:3:2Front	3@0	3@3	2@6	0
6	0	3:3:2Back	3@8	3@11	2@14	0
7	0	Reserved	8@0	8@8	8@16	8@24
8	0	Reserved	8@0	8@8	8@16	8@24
9	0	2:3:2:1Front	2@0	3@2	2@5	1@7
10	0	2:3:2:1Back	2@8	3@10	2@13	1@15
11	0	2:3:2FrontOff	2@0	3@2	2@5	0
12	0	2:3:2BackOff	2@8	3@10	2@13	0
13	0	5:5:5:1Back	5@16	5@21	5@26	1@31
14	0	CI8	-	-	-	-
15	0	Reserved	8@0	8@8	8@16	8@24
16	0	5:6:5Front	5@0	6@5	5@11	0
17	0	5:6:5Back	5@16	6@21	5@27	0
18	0	Reserved	8@0	8@8	8@16	8@24
19..31	0	Reserved	8@0	8@8	8@16	8@24
0	1	8:8:8:8	8@16	8@8	8@0	8@24
1	1	5:5:5:1Front	5@10	5@5	5@0	1@15
2	1	4:4:4:4	4@8	4@4	4@0	4@12
3	1	Reserved	8@16	8@8	8@0	8@24
4	1	Reserved	8@16	8@8	8@0	8@24
5	1	3:3:2Front	3@5	3@2	2@0	0
6	1	3:3:2Back	3@13	3@10	2@8	0
7	1	Reserved	8@16	8@8	8@0	8@24
8	1	Reserved	8@16	8@8	8@0	8@24
9	1	2:3:2:1Front	2@5	3@2	2@0	1@7
10	1	2:3:2:1Back	2@13	3@10	2@8	1@15
11	1	2:3:2FrontOff	2@5	3@2	2@0	0
12	1	2:3:2BackOff	2@13	3@10	2@8	0
13	1	5:5:5:1Back	5@26	5@21	5@16	1@31
14	1	CI8	-	-	-	-
15	1	Reserved	8@16	8@8	8@0	8@24
16	1	5:6:5Front	5@11	6@5	5@0	0
17	1	5:6:5Back	5@27	6@21	5@16	0
19..31	1	Reserved	8@16	8@8	8@0	8@24

RDCursorControl

Name	Type	Offset	Format
RDCursorControl	RAMDAC Control *Control register*	0x006	Bitfield

Bits	Name	Read	Write	Reset	Description
0	DoubleX	✓	✓	0	0 = Disabled. 1 = Enabled.
1	DoubleY	✓	✓	0	0 = Disabled. 1 = Enabled.
2	ReadbackPosition	✓	✓	0	0 = Disabled - readback last value written. 1 = Enabled - readback position in use.
3..7	Reserved	✓	×	0	

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register

RDCursorHotSpotX

Name	Type	Offset	Format
RDCursorHotSpotX	RAMDAC Control *Control register*	0x00B	Integer

Bits	Name	Read	Write	Reset	Description
0..5	X	✓	✓	X	X position of hot spot in cursor.
6..7	Reserved	✓	×	0	

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register

RDCursorHotSpotY

Name	Type	Offset	Format
RDCursorHotSpotY	RAMDAC Control *Control register*	0x00C	Integer

Bits	Name	Read	Write	Reset	Description
0..5	Y	✓	✓	X	Y position of hot spot in cursor.
6..7	Reserved	✓	×	0	

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register

RDCursorMode

Name	Type	Offset	Format
RDCursorMode	RAMDAC Control *Control register*	0x005	Bitfield

Bits	Name	Read	Write	Reset	Description
0	CursorEnable	✓	✓	0	0 = Disabled. 1 = Enabled.
1..3	Format	✓	✓	0	0 = 64x64 (2 bits per entry, partitions 0, 1, 2, and 3). 1 = 32x32 (2 bits per entry, partition 0). 2 = 32x32 (2 bits per entry, partition 1). 3 = 32x32 (2 bits per entry, partition 2). 4 = 32x32 (2 bits per entry, partition 3). 5 = 32x32 (4 bits per entry, partitions 0 and 1). 6 = 32x32 (4 bits per entry, partitions 2 and 3).
4..5	Type	✓	✓	0	0 = Microsoft Windows. 1 = X Windows 2 = 3 Color 3 = 15 color
6	ReversePixelOrder	✓	✓	0	0 = Disabled (incrementing pixel index goes left to right on screen). 1 = Enabled (incrementing pixel index goes right to left on screen).
7	Reserved	✓	✗	0	

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the *RDIndexedData* register

RDCursorPalette[0…44]

Name	Type	Offset	Format
RDCursorPalette[0…44]	RAMDAC Control *Control register*	0x303 to 0x32F	Integer

Bits	Name	Read	Write	Reset	Description
0..7	Color	✓	✓	X	Stores the red, green, and blue color components for 15 cursor colors. These index from 1 to 15.

Notes: These registers are accessed indirectly by first loading the indexes into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.

RDCursorPattern[0…1023]

Name	Type	Offset	Format
RDCursorPattern[0…1023]	RAMDAC Control *Control register*	0x400 to 0x7FF	Integer

Bits	Name	Read	Write	Reset	Description
0..7	Pattern	✓	✓	X	Bitmap for the cursor

Notes: These registers are accessed indirectly by first loading the indexes into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register

RDCursorXHigh

Name	Type	Offset	Format
RDCursortXHigh	RAMDAC Control *Control register*	0x008	Integer

Bits	Name	Read	Write	Reset	Description
0..3	XHigh	✓	✓	X	The high order bits of the cursor X position.
4..7	Reserved	✓	×	0	

Notes:
1. This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.
2. Value at readback is determined by the ReadbackPosition field in the RDCursorControl register.

RDCursorXLow

Name	Type	Offset	Format
RDCursortXLow	RAMDAC Control *Control register*	0x007	Integer

Bits	Name	Read	Write	Reset	Description
0..7	XLow	✓	✓	X	The low order bits of the cursor X position.

Notes:
1. This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.
2. Value at readback is determined by the ReadbackPosition field in the RDCursorControl register

RDCursorYHigh

Name	Type	Offset	Format
RDCursorYHigh	RAMDAC Control *Control register*	0x00A	Integer

Bits	Name	Read	Write	Reset	Description
0..3	YHigh	✓	✓	X	The high order bits of the cursor Y position.
4..7	Reserved	✓	✗	0	

Notes: 1. This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.
2. Value at readback is determined by the ReadbackPosition field in the RDCursorControl register.

RDCursorYLow

Name	Type	Offset	Format
RDCursorYLow	RAMDAC Control *Control register*	0x009	Integer

Bits	Name	Read	Write	Reset	Description
0..7	YLow	✓	✓	X	The low order bits of the cursor Y position.

Notes: 1. This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.
2. Value at readback is determined by the ReadbackPosition field in the RDCursorControl register.

RDDACControl

Name	Type	Offset	Format
RDDACControl	RAMDAC Control *Control register*	0x002	Bitfield

Bits	Name	Read	Write	Reset	Description
0..2	DACPowerCtl	✓	✓	0	0 = Normal operation. 1 = LowPower
3	Reserved	✓	✓	0	[SyncOnGreen]

4	BlankRedDAC	✓	✓	0	0 = Disabled. 1 = Enabled.
5	BlankGreen DAC	✓	✓	0	0 = Disabled. 1 = Enabled.
6	BlankBlueDAC	✓	✓	0	0 = Disabled. 1 = Enabled.
7	BlankPedestal	✓	✓	0	0 = Disabled. 1 = Enabled.

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.

RDDClk0FeedbackScale

Name	Type	Offset	Format
RDDClk0FeedbackScale	RAMDAC Control *Control register*	0x202	Integer

Bits	Name	Read	Write	Reset	Description
0..7	Value	✓	✓	0x7	

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register

RDDClk0PostScale

Name	Type	Offset	Format
RDDClk0PostScale	RAMDAC Control *Control register*	0x203	Integer

Bits	Name	Read	Write	Reset	Description
0..2	Scale	✓	✓	0	0 = Divide by 1. 1 = Divide by 2. 2 = Divide by 4. 3 = Divide by 8. 4 = Divide by 16 5..7 = Reserved
3..7	Reserved	✓	✗	0	

Notes: This register is accessed indirectly by first loading the index into the *RDIndexLow* and *RDIndexHigh* registers, and then reading or writing the *RDIndexedData* register.

RDDClk1PostScale
RDDClkPostScale

Name	Type	Offset	Format
RDDClk1PostScale	RAMDAC Control	0x206	Integer
RDKClkPostScale	RAMDAC Control	0x210	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..2	Scale	✓	✓	X	0 = Divide by 1. 1 = Divide by 2 2 = Divide by 4. 3 = Divide by 8. 4 = Divide by 16. 5..7 = Reserved
3..7	Reserved	✓	×	0	

Notes: This register is accessed indirectly by first loading the index into the *RDIndexLow* and *RDIndexHigh* registers, and then reading or writing the *RDIndexedData* register.

RDDClk2PostScale
RDDClk3PostScale

Name	Type	Offset	Format
RDDClk2PostScale	RAMDAC Control	0x209	Integer
RDDClk3PostScale	RAMDAC Control	0x20C	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..2	Scale	✓	✓	X	0 = Divide by 1. 1 = Divide by 2. 2 = Divide by 4. 3 = Divide by 8.
					4 = Divide by 16. 5..7 = Reserved
3..7	Reserved	✓	×	0	

Notes: This register is accessed indirectly by first loading the index into the *RDIndexLow* and *RDIndexHigh* registers, and then reading or writing the *RDIndexedData* register.

RDDClk0PreScale

Name	Type	Offset	Format
RDDClk0PreScale	RAMDAC Control *Control register*	0x201	Integer

Bits	Name	Read	Write	Reset	Description
0..7	Value	✓	✓	0x4	

Notes: This register is accessed indirectly by first loading the index into the *RDIndexLow* and *RDIndexHigh* registers, and then reading or writing the *RDIndexedData* register.

RDDClk1FeedbackScale

Name	Type	Offset	Format
RDDClk1FeedbackScale	RAMDAC Control *Control register*	0x24F	Integer

Bits	Name	Read	Write	Reset	Description
0..7	Value	✓	✓	0x4F	

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.

RDDClk1PreScale

Name	Type	Offset	Format
RDDClk1PreScale	RAMDAC Control *Control register*	0x28	Integer

Bits	Name	Read	Write	Reset	Description
0..7	Value	✓	✓	0x28	

Notes: This register is accessed indirectly by first loading the index into the *RDIndexLow* and *RDIndexHigh* registers, and then reading or writing the *RDIndexedData* register.

RDDClk2FeedbackScale
RDDClk3FeedbackScale

Name	Type	Offset	Format
RDDClk2FeedbackScale	RAMDAC Control	0x208	Integer
RDDClk3FeedbackScale	RAMDAC Control	0x20B	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..7	Value	✓	✓	X	

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register

RDDClk2PreScale
RDDClk3PreScale

Name	Type	Offset	Format
RDDClk2PreScale	RAMDAC Control	0x207	Integer
RDDClk3PreScale	RAMDAC Control	0x20A	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..7	Value	✓	✓	X	

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register

RDDClkControl

Name	Type	Offset	Format
RDDClkControl	RAMDAC Control *Control register*	0x200	bitfield

Bits	Name	Read	Write	Reset	Description	
0	Clock	✓	✓	1	0 = Disable	1 = Enable
1	Lock	✓	✗	X	0 = Not locked.	1 = Locked.
2..3	State	✓	✓	0x2	0 = Drive Low 2 = Run	1 = Drive High 3 = Reserved
4..5	Source	✓	✓	0	0 = PLL 2 = VideoStreamB	1 = VideoStreamA 3 = External
6..7	Reserved	✓	✗	0		

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register

RDDClkSetup1
RDKClkSetup1

Name	Type	Offset	Format
RDDClkSetup1	RAMDAC Control	0x1F0	Integer
RDKClkSetup1	RAMDAC Control *Control register*	0x1F2	Integer

Bits	Name	Read	Write	Reset	Description
0..7	Setup	✓	✓	0x1C	

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.

RDDClkSetup2
RDKClkSetup2

Name	Type	Offset	Format
RDDClkSetup2	RAMDAC Control	0x1F1	Integer
RDKClkSetup2	RAMDAC Control *Control register*	0x1F3	Integer

Bits	Name	Read	Write	Reset	Description
0	Setup	✓	✓	1	
1..7	Reserved	✓	✗	0	

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register

RDKClkControl

Name	Type	Offset	Format
RDKclkControl	RAMDAC Control *Control register*	0x20D	Bitfield

Bits	Name	Read	Write	Reset	Description
0	Clock	✓	✓	1	0 = Disable 1 = Enable
1	Lock	✓	✗	0	0 = NotLocked 1 = Locked
2..3	State	✓	✓	0x2	0 = Drive Low 1 = Drive High 2 = Run 3 = Low Power
4..6	Source	✓	✓	0	0 = PClk 1 = PClk/2 2 = PLL 3..7 = Reserved
7	Reserved	✓	✗	0	

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.

RDKClkFeedbackScale

Name	Type	Offset	Format
RDKClkFeedbackScale	RAMDAC Control *Control register*	0x20F	Integer

Bits	Name	Read	Write	Reset	Description
0..7	Value	✓	✓	0x20	

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register

RDKClkPreScale

Name	Type	Offset	Format
RDKClkPreScale	RAMDAC Control *Control register*	0x20E	Integer

Bits	Name	Read	Write	Reset	Description
0..7	Value	✓	✓	0x10	

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register

RDMClkControl

Name	Type	Offset	Format
RDMClkControl	RAMDAC Control Command register	0x211	Bitfield

Bits	Name	Read	Write	Reset	Description
0	Clock	✓	✓	1	0 = Disable 1 = Enable
1	Reserved	✓	✗	0	
2..3	State	✓	✓	0x2	0 = Drive Low 1 = Drive High 2 = Run 3 = Low Power
4..6	Source	✓	✓	0x2	0 = PClk 1 = PClk/2 2 = Reserved 3 = ExternalMClk/2 4 = ExternalMClk 5 = KClk PLL/2 6 = KClk PLL 7 = Reserved
7	Reserved	✓	✗	0	

Notes: This register is accessed indirectly by first loading the index into the **RDIndexLow** and RDIndexHigh registers, and then reading or writing the **RDIndexedData** register.
When sourcing from KClk (Source=5 or Source=6) note that the KClk value is always set to the PLL, not to the value determined by the **KclkControl** register.

RDMiscControl

Name	Type	Offset	Format
RDMiscControl	RAMDAC Control Command register	0x000	Bitfield

Bits	Name	Read	Write	Reset	Description
0	HighColor Resolution	✓	✓	0	Controls the width of the palette data. 0 = Disabled - use 6 bits per entry. 1 = Enabled - use 8 bits per entry.
1	PixelDouble	✓	✓	0	0 = Disabled. 1 = Enabled.
2	LastRead Address	✓	✓	0	Controls data returned by read from RDPaletteReadAddress register. 0 = Disabled - return palette access state. 1 = Enabled - return last palette read address.
3	DirectColor	✓	✓	0	0 = Disabled. 1 = Enabled.
4	Overlay	✓	✓	0	0 = Disabled. 1 = Enabled.
5	PixelDouble Buffer	✓	✓	0	0 = Disabled. 1 = Enabled.
6	VSBOutput	✓	✓	0	Video Stream Port B Output
					0 = Disabled 1 = Enabled
7	StereoDouble Buffer	✓	✓	0	Controls per-pixel double buffering in 5551 color format. 0 = Disabled. 1 = Enabled.

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.

RDOverlayKey

Name	Type	Offset	Format
RDOverlayKey	RAMDAC Control *Control register*	0x00D	Integer

Bits	Name	Read	Write	Reset	Description
0..7	Key	✓	✓	X	Indicates the overlay bit pattern that should be treated as transparent.

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register

RDPaletteData

Name	Type	Offset	Format
RDPaletteData	RAMDAC Control *Control register*	0x4008	Integer

Bits	Name	Read	Write	Reset	Description
0..7	Data	✓	✓	X	

Notes:
1. If the color resolution is 6 bits, bits 6 and 7 are returned as zero for reads and ignored for writes. In this mode, bits 0 to 5 are read from, or written to, bits 2 to 7 of the palette. A read auto-increments RDPaletteReadAddress and RDPaletteWriteAddress, whereas a write autoincrements the RDPallettWriteAddress only.
2. The register is accessed directly by reading or writing to the defined address. It is a byte wide and set on an 8 byte boundary in the PCI address range. When accessed from the SVGA it is set on a byte boundary.

RDPaletteReadAddress

Name	Type	Offset	Format
RDPaletteReadAddress	RAMDAC Control *Control register*	0x4018	Integer

Bits	Name	Read	Write	Reset	Description
0..7	Address	✓	✓	X	

Notes: The register is accessed directly by reading or writing to the defined address. It is a byte wide and set on an 8 byte boundary in the PCI address range. When accessed from the SVGA it is set on a byte boundary.

RDPaletteWriteAddress

Name	Type	Offset	Format
RDPaletteWriteAddress	RAMDAC Control *Control register*	0x4000	Integer

Bits	Name	Read	Write	Reset	Description
0..7	Address	✓	✓	0	

Notes: The register is accessed directly by reading or writing to the defined address. It is a byte wide and set on an 8 byte boundary in the PCI address range. When accessed from the SVGA it is set on a byte boundary.

RDPan

Name	Type	Offset	Format
RDPan	RAMDAC Control *Control register*	0x00E	Bitfield

Bits	Name	Read	Write	Reset	Description
0	Enable	✓	✓	X	Delay data by 32 bits.
1	Gate	✓	✓	X	Discard first 64 bits on line.
7..2	Reserved	✓	✕	X	

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.

RDPixelMask

Name	Type	Offset	Format
RDPixelMask	RAMDAC Control *Control register*	0x4010	Integer

Bits	Name	Read	Write	Reset	Description
0..7	Mask	✓	✓	X	

Notes:
1. The contents of this register is ANDed with the index into the color palette. The same mask is applied separately to red, green, and blue components.
2. The register is accessed directly by reading or writing to the defined address. It is a byte wide and set on an 8 byte boundary in the PCI address range. When accessed from the SVGA it is set on a byte boundary

RDPixelSize

Name	Type	Offset	Format
RDPixelSize	RAMDAC Control *Control register*	0x003	Integer

Bits	Name	Read	Write	Reset	Description
0..2	Pixel Size	✓	✓	X	0 = 8 bits. 1 = 16 bits. 2 = 32 bits. 3 = Reserved 4 = 24 bits. 5..7 = Reserved
3..7	Reserved	✓	×	0	

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register

RDSClkControl

Name	Type	Offset	Format
RDSClkControl	RAMDAC Control *Control register*	0x215	Bitfield

Bits	Name	Read	Write	Reset	Description
0	Clock	✓	✓	1	0 = Disable 1 = Enable
1	Reserved	✓	×	0	
2..3	State	✓	✓	0x2	0 = Drive Low 1 = Drive High 2 = Run 3 = Low Power
4..6	Source	✓	✓	0x0	0 = PClk 1 = PClk/2 2 = Reserved 3 = ExternalSClk/2 4 = ExternalSClk 5 = KClk/2 6 = KClk 7 = Reserved
7	Reserved	✓	×	0	

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register

RDScratch

Name	Type	Offset	Format
RDScratch	RAMDAC Control *Control register*	0x001F	Integer

Bits	Name	Read	Write	Reset	Description
0..7	Scratch	✓	✓	X	User definable register for storing state.

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.

RDSense

Name	Type	Offset	Format
RDSense	RAMDAC Control *Control register*	0x00F	Bitfield

Bits	Name	Read	Write	Reset	Description
0	Red	✓	×	X	
1	Green	✓	×	X	
2	Blue	✓	×	X	
3..7	Reserved	✓	×	0	

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register

RDSyncControl

Name	Type	Offset	Format
RDSyncControl	RAMDAC Control *Control register*	0x001	Bitfield

Bits	Name	Read	Write	Reset	Description
0..2	HSyncCtl	✓	✓	0	0 = Active low at pin. 1 = Active high at pin. 2 = Tri-state at pin. 3 = Force active 5..7 = Reserved
3..5	VSyncCtl	✓	✓	0	0 = Active low at pin. 1 = Active high at pin. 2 = Tri-state at pin. 3 = Force active. 4 = Force inactive. 5..7 = Reserved
6	HSyncOverride	✓	✓	0	0 = As set by HsyncCtl 1 = Force high
7	VSyncOverride	✓	✓	0	0 = As set by VsyncCtl1 = Force high

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.

Decimal values for **MSBs used**
0 = 0%
64 = 25%
128 = 50%
192 = 75%

RDVideoOverlayBlend

Name	Type	Offset	Format
RDVideoOverlayBlend	RAMDAC Control *Control register*	0x002C	Integer

Bits	Name	Read	Write	Reset	Description
0..5	Reserved	✓	✗	0	
6..7	Factor	✓	✓	X	Proportion to blend main image and overlay, enabled by BlendSrc field of RDVideoOverlay Control Field register. 0 = 0% 0x1 = 25% 0x2 = 59% 0x3 = 75%

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.

RDVideoOverlayControl

Name	Type	Offset	Format
RDVideoOverlayControl	RAMDAC Control *Control register*	0x020	Bitfield

Bits	Name	Read	Write	Reset	Description
0	Enable	✓	✓	0	0 = Disabled. 1 = Enabled.
1..2	Mode	✓	✓	X	0 = MainKey 1 = OverlayKey
					2 = Always 3 = Blend
3	DirectColor	✓	✓	X	0 = Disabled. 1 = Enabled.
4	BlendSrc	✓	✓	X	0 = Main. 1 = Register.
5	Key	✓	✓	X	0 = Color. 1 = Alpha.
6..7	Reserved	✓	✗	0	

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.

RDVideoOverlayKeyB

Name	Type	Offset	Format
RDVideoOverlayKeyB	RAMDAC Control *Control register*	0x02B	Integer

Bits	Name	Read	Write	Reset	Description
0..7	Blue	✓	✓	X	The blue component for color key checking

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register

RDVideoOverlayKeyG

Name	Type	Offset	Format
RDVideoOverlayKeyG	RAMDAC Control *Control register*	0x02A	Integer

Bits	Name	Read	Write	Reset	Description
0..7	Green	✓	✓	X	The green component for color key checking

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.

RDVideoOverlayKeyR

Name	Type	Offset	Format
RDVideoOverlayKeyR	RAMDAC Control *Control register*	0x029	Integer

Bits	Name	Read	Write	Reset	Description
0..7	Red	✓	✓	X	The red component for color key checking is also used to hold the alpha value during alpha test.

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register

RDVideoOverlayXEndHigh

Name	Type	Offset	Format
RDVideoOverlayXEndHigh	RAMDAC Control *Control register*	0x026	Integer

Bits	Name	Read	Write	Reset	Description
0..3	XEndHigh	✓	✓	X	High order bits of right hand edge of video overlay.
4..7	Reserved	✓	×	0	

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.

RDVideoOverlayXEndLow

Name	Type	Offset	Format
RDVideoOverlayXEndLow	RAMDAC Control *Control register*	0x025	Integer

Bits	Name	Read	Write	Reset	Description
0..7	XEndLow	✓	✓	X	Low order bits of right hand edge of video overlay.

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register

RDVideoOverlayXStart High

Name	Type	Offset	Format
RDVideoOverlayXStart High	RAMDAC Control *Control register*	0x022	Integer

Bits	Name	Read	Write	Reset	Description
0..3	XStartHigh	✓	✓	X	High order bits of left hand edge of video overlay.
4..7	Reserved	✓	✗	0	

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.

RDVideoOverlayXStartLow

Name	Type	Offset	Format
RDVideoOverlayXStartLow	RAMDAC Control ***Control register***	0x021	Integer

Bits	Name	Read	Write	Reset	Description
0..7	XStartLow	✓	✓	X	Low order bits of left hand edge of video overlay.

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.

RDVideoOverlayYEndHigh

Name	Type	Offset	Format
RDVideoOverlayYEndHigh	RAMDAC Control *Control register*	0x028	Integer

Bits	Name	Read	Write	Reset	Description
0..3	YEndHigh	✓	✓	X	High order bits of last line of video overlay.
4..7	Reserved	✓	✗	0	

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.

RDVideoOverlayYEndLow

Name	Type	Offset	Format
RDVideoOverlayYEndLow	RAMDAC Control *Control register*	0x027	Integer

Bits	Name	Read	Write	Reset	Description
0..7	YEndLow	✓	✓	X	Low order bits of last line of video overlay.

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.

RDVideoOverlayYStartHigh

Name	Type	Offset	Format
RDVideoOverlayYStartHigh	RAMDAC Control *Control register*	0x024	Integer

Bits	Name	Read	Write	Reset	Description
0..3	YStartHigh	✓	✓	X	High order bits of first line of video overlay.
4..7	Reserved	✓	✕	0	

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register.

RDVideoOverlayYStartLow

Name	Type	Offset	Format
RDVideoOverlayYStartLow	RAMDAC Control *Control register*	0x023	Integer

Bits	Name	Read	Write	Reset	Description
0..7	YStartLow	✓	✓	X	Low order bits of first line of video overlay.

Notes: This register is accessed indirectly by first loading the index into the RDIndexLow and RDIndexHigh registers, and then reading or writing the RDIndexedData register

4.8 Region 0 Video Stream Processing (0x5000-0x5FFF)

VSAControl

Name	Type	Offset	Format
VSAControl	Video stream Control *Control register*	0x5900	Bitfield

Bits	Name	Read	Write	Reset	Description	
0	Video	✓	✓	0	0 = Disable	1 = Enable
1	VBI	✓	✓	0	0 = Disable	1 = Enable
2	BufferCtl	✓	✓	0	0 = Double buffered	1 = Triple buffered
3..4	ScaleX	✓	✓	0	0 = 1:1 2 = 4:1	1 = 2:1 3 = 8:1
5..6	ScaleY	✓	✓	0	0 = 1:1 2 = 4:1	1 = 2:1 3 = 8:1
7	MirrorX	✓	✓	0	0 = Disable	1 = Enable
8	MirrorY	✓	✓	0	0 = Disable	1 = Enable
9..10	Discard	✓	✓	0	0 = None 2 = FieldTwo	1 = FieldOne 3 = Reserved
11	CombineFields	✓	✓	0	0 = Disable	1 = Enable
12	LockTo StreamB	✓	✓	0	0 = Disable	1 = Enable
13	Patch	✓	✓	0	0 = Disable	1 = Enable
14..19	PatchOffsetX	✓	✓	0		
20..23	PatchOffsetY	✓	✓	0		
24..25	PixelSize	✓	✓	0	0 = 1 byte 2 = 4 bytes	1 = 2 bytes 3 = Reserved
26	LockToVideoOverlay	✓	✓	0	0 = Disable	1 = Enable
27	LockToVideo	✓	✓	0	0 = Disable	1 = Enable
28..31	Reserved	✓	✗	0		

Notes:

VSACurrentLine

Name	Type	Offset	Format
VSACurrentLine	Video stream Control	0x5910	Integer
VSBCurrentLine	Video stream Control	0x5A10	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..10	Line	✓	×	X	Current line number, reference to start of VRef.
11..31	Reserved	✓	×	0	

Notes:

VSADroppedFrames

Name	Type	Offset	Format
VSADroppedFrames	Video stream Control	0x59D8	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..7	Count	✓	✓ (to reset)	0	Count of dropped frames
8..31	Reserved	✓	×	0	

Notes:

VSAFifoControl

Name	Type	Offset	Format
VSAFifoControl	Video stream Control	0x59B8	Bitfield
VSBFifoControl	Video stream Control	0x5AB8	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0..7	LP Threshold	✓	✓	0x8	Low Priority Threshold
8...15	HP Threshold	✓	✓	0x8	High Priority Threshold
16..31	Reserved	✓	✗	0	

VSAInterruptLine

Name	Type	Offset	Format
VSAInterruptLine	Video stream Control	0x5908	Integer
VSBInterruptLine	Video stream Control	0x5A08	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..10	Line	✓	✓	X	Line number to generate interrupt.
11..31	Reserved	✓	✗	0	

Notes:

VSATimeStamp0

Name	Type	Offset	Format
VSATimeStamp0	Video stream Control	0x59C0	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..31	Time	✓	✗	0	Capture time of buffer 0

Notes:

VSATimeStamp1

Name	**Type**	**Offset**	**Format**
VSATimeStamp1	Video stream Control *Control register*	0x59C8	Integer

Bits	Name	Read	Write	Reset	Description
0..31	Time	✓	✕	0	Capture time of buffer 1

Notes:

VSATimeStamp2

Name	**Type**	**Offset**	**Format**
VSATimeStamp2	Video stream Control ***Control register***	0x59D0	Integer

Bits	Name	Read	Write	Reset	Description
0..31	Time	✓	✕	0	Capture time of buffer 2

Notes:

VSAVBIAddress0

Name	Type	Offset	Format
VSAVBIAddress0	Video stream Control	0x5978	Integer
VSAVideoAddress0	Video stream Control	0x5928	Integer
VSBVBIAddress0	Video stream Control	0x5A78	Integer
VSBVideoAddress0	Video stream Control *Control register*	0x5A28	Integer

Bits	Name	Read	Write	Reset	Description
0..20	Base	✓	✓	X	Base address (128 bit aligned)
21..31	Reserved	✓	✗	0	

Notes:

VSAVBIAddress1

Name	Type	Offset	Format
VSAVBIAddress1	Video stream Control	0x5980	Integer
VSAVideoAddress1	Video stream Control	0x5930	Integer
VSBVBIAddress1	Video stream Control	0x5A80	Integer
VSBVideoAddress1	Video stream Control *Control register*	0x5A30	Integer

Bits	Name	Read	Write	Reset	Description
0..20	Base	✓	✓	X	Base address (128 bit aligned)
21..31	Reserved	✓	✗	0	

Notes:

VSAVBIAddress2

Name	Type	Offset	Format
VSAVBIAddress2	Video stream Control	0x5988	Integer
VSAVideoAddress2	Video stream Control	0x5938	Integer
VSBVBIAddress2	Video stream Control	0x5A88	Integer
VSBVideoAddress2	Video stream Control *Control register*	0x5A38	Integer

Bits	Name	Read	Write	Reset	Description
0..20	Base	✓	✓	X	Base address (64 bit aligned)
21..31	Reserved	✓	✗	0	

Notes:

VSAVBIAddressHost

Name	Type	Offset	Format
VSAVBIAddressHost	Video stream Control	0x5968	Integer
VSBVBIAddressHost	Video stream Control *Control register*	0x5A68	Integer

Bits	Name	Read	Write	Reset	Description
0..1	Base	✓	✓	X	Base address register index
2..31	Reserved	✓	✗	0	

Notes:

VSAVBIAddressIndex

Name	Type	Offset	Format
VSAVBIAddressIndex	Video stream Control	0x5970	Integer
VSAVideoAddressIndex	Video stream Control	0x5920	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..1	Base	✓	✗	0	Base address register index
2..31	Reserved	✓	✗	0	

Notes:

VSAVBIEndData

Name	Type	Offset	Format
VSAVBIEndData	Video stream Control	0x59B0	Integer
VSBVBIEndData	Video stream Control	0x5AB0	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..10	First Clock	✓	✓	X	First clock after VBI data
11..31	Reserved	✓	✗	0	

Notes:

VSAVBIEndLine

Name	Type	Offset	Format
VSAVBIEndLine	Video stream Control	0x59A0	Integer
VSBVBIEndLine	Video stream Control	0x5AA0	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..10	First Line	✓	✓	X	First scanline after VBI data
11..31	Reserved	✓	×	0	

Notes:

VSAVBIStartData

Name	Type	Offset	Format
VSAVBIStartData	Video stream Control	0x59A8	Integer
VSBVBIStartData	Video stream Control	0x5AA8	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..10	First Data	✓	✓	X	First valid data in VBI line.
11..31	Reserved	✓	×	0	

Notes:

VSAVBIStartLine

Name	Type	Offset	Format
VSAVBIStartLine	Video stream Control	0x5998	Integer
VSBVBIStartLine	Video stream Control	0x5A98	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..10	First Line	✓	✓	X	First scanline of VBI data
11..31	Reserved	✓	×	0	

Notes:

VSAVBIStride

Name	Type	Offset	Format
VSAVBIStride	Video stream Control	0x5990	Integer
VSAVideoStride	Video stream Control	0x5940	Integer
VSBVBIStride	Video stream Control	0x5A90	Integer
VSBVideoStride	Video stream Control	0x5A40	Integer
	Control register		

0..20	Stride	✓	✓	X	Stride between scanlines (in 128 bit units).
21..31	Reserved	✓	×	0	

Notes:

VSAVideoAddress2 see VSAVBIAddress2

VSAVideoAddress1 see VSAVBIAddress1

VSAVideoAddress0 see VSAVBIAddress0

VSAVBIAddress0

Name	Type	Offset	Format
VSAVBIAddress0	Video stream Control	0x5978	Integer
VSAVideoAddress0	Video stream Control	0x5928	Integer
VSBVBIAddress0	Video stream Control	0x5A78	Integer
VSBVideoAddress0	Video stream Control *Control register*	0x5A28	Integer

Bits	Name	Read	Write	Reset	Description
0..20	Base	✓	✓	X	Base address (128 bit aligned)
21..31	Reserved	✓	×	0	

Notes:

VSAVBIAddress1

Name	Type	Offset	Format
VSAVBIAddress1	Video stream Control	0x5980	Integer
VSAVideoAddress1	Video stream Control	0x5930	Integer
VSBVBIAddress1	Video stream Control	0x5A80	Integer
VSBVideoAddress1	Video stream Control *Control register*	0x5A30	Integer

Bits	Name	Read	Write	Reset	Description
0..20	Base	✓	✓	X	Base address (128 bit aligned)
21..31	Reserved	✓	×	0	

Notes:

VSAVBIAddress2

Name	Type	Offset	Format
VSAVBIAddress2	Video stream Control	0x5988	Integer
VSAVideoAddress2	Video stream Control	0x5938	Integer
VSBVBIAddress2	Video stream Control	0x5A88	Integer
VSBVideoAddress2	Video stream Control	0x5A38	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..20	Base	✓	✓	X	Base address (64 bit aligned)
21..31	Reserved	✓	✗	0	

Notes:

VSAVideoAddressHost

Name	Type	Offset	Format
VSAVideoAddressHost	Video stream Control	0x5918	Integer
VSBVideoAddressHost	Video stream Control	0x5A18	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..1	Host base	✓	✓	X	Host base address register index
2..31	Reserved	✓	✗	0	

Notes:

VSAVideoAddressIndex see VSAVBIAddressIndex

VSAVideoEndData

Name	Type	Offset	Format
VSAVideoEndData	Video stream Control	0x5960	Integer
VSBVideoEndData	Video stream Control *Control register*	0x5A60	Integer

Bits	Name	Read	Write	Reset	Description
0..10	First Clock	✓	✓	X	First clock after active video
11..31	Reserved	✓	✗	0	

Notes:

VSAVideoEndLine

Name	Type	Offset	Format
VSAVideoEndLine	Video stream Control	0x5950	Integer
VSBVideoEndLine	Video stream Control *Control register*	0x5A50	Integer

Bits	Name	Read	Write	Reset	Description
0..10	First Line	✓	✓	X	First scanline after Video data
11..31	Reserved	✓	✗	0	

Notes:

VSAVideoStartData

Name	Type	Offset	Format
VSAVideoStartData	Video stream Control	0x5958	Integer
VSBVideoStartData	Video stream Control	0x5A58	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..10	First Data	✓	✓	X	First valid data in video line.
11..31	Reserved	✓	×	0	

Notes:

VSAVideoStartLine

Name	Type	Offset	Format
VSAVideoStartLine	Video stream Control	0x5948	Integer
VSBVideoStartLine	Video stream Control	0x5A48	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..10	First Line	✓	✓	X	First scanline of video data
11..31	Reserved	✓	×	0	

Notes:

VSAVideoStride see VSAVBIAddress0

VSBControl

Name	Type	Offset	Format
VSBControl	Video stream Control *Control register*	0x5A00	Bitfield

Bits	Name	Read	Write	Reset	Description	
0	Video	✓	✓	0	0 = Disable	1 = Enable
1	VBI	✓	✓	0	0 = Disable	1 = Enable
2	BufferCtl	✓	✓	0	0 = Double buffered	1 = Triple buffered
3	CombineFields	✓	✓	0	0 = Disable	1 = Enable
8..4	ColorFormat	✓	✓	0		
9..10	PixelSize	✓	✓	0	0 = 1 byte 2 = 4 bytes	1 = 2 bytes 3 = Reserved
11	RGB Order	✓	✓	0	0 = BGR	1 = RGB
12	GammaCorrect	✓	✓	0	0 = Disable	1 = Enable
13	LockTo StreamA	✓	✓	0	0 = Disable	1 = Enable
14	RAMDAC	✓	✓	0	0 = Disable	1 = Enable
15	Patch	✓	✓	0	0 = Disable	1 = Enable
16..21	PatchOffsetX	✓	✓	0		
22..25	PatchOffsetY	✓	✓	0		
26	LockToOverlay	✓	✓	0	0 = Disable	1 = Enable
27	LockToVideo	✓	✓	0	0 = Disable	1 = Enable
28..31	Reserved	✓	×	0		

Notes:

VSBCurrentLine see VSACurrentLine

VSBFifoControl see VSAFIFOControl

VSBInterruptLine see VSAInterruptLine

VSBVBIAddress0 see VSAVBIAddress0

VSBVBIAddress1 see VSAVBIAddress1

VSBVBIAddress2 see VSAVBIAddress2

VSBVBIAddressHost see VSAVBIAddressHost

VSBVBIAddressIndex

Name	Type	Offset	Format
VSBVBIAddressIndex	Video stream Control	0x5A70	Integer
VSBVideoAddressIndex	Video stream Control *Control register*	0x5A20	Integer

Bits	Name	Read	Write	Reset	Description
0..1	Base	✓	✗	0x2	Base address register index
2..31	Reserved	✓	✗	0x2	

VSBVBIEndData see VSAVBIEndData

VSBVBIEndLine see VSAVBIEndLine

VSBVBIStartData see VSAVBIStartData

VSBVBIStartLine see VSAVBIStartLine

VSBVBIStride see VSAVBIStride

VSBVideoAddress0 see VSAVBIAddress0

VSBVideoAddress1 see VSAVBIAddress1

VSBVideoAddress2 see VSAVBIAddress2

VSBVideoAddressHost see VSAVideoAddressHost

VSBVideoAddressIndex see VSBVBIAddressIndex

VSBVideoEndData see VSAVideoEndData

VSBVideoEndLine see VSAVideoEndLine

VSBVideoStartData see VSAVideoStartData

VSBVideoStartLine see VSAVideoStartline

VSBVideoStride see VSAVBIStride

VSConfiguration

Name	Type	Offset	Format
VSConfiguration	Video stream Control *Control register*	0x5800	Bitfield

Bits	Name	Read	Write	Reset	Description
0..2	Unit mode	✓	✓	0	0 = ROM Access 1 = MPEG data to decoder via GP bus, decoded video into input port. 2 = Wide output 16 bit. 3 = Simultaneous input and output, program decoder and encoder through I2C. 4 = Wide input 16 bit. 5 = VSA/VSB reset removed, use to probe for external chips. 6 = Drive flat panels 7 = Default to mode 0.
3	GPModeA	✓	✓	0	0 = Operate GP bus in Mode B 1 = Operate GP bus in Mode A
4	VActiveVideoA	✓	✓	1	0 = Ignore VActive for Video data 1 = Gate Video data with VActive
5	VActiveVideoB	✓	✓	1	0 = Ignore VActive for Video data 1 = Gate Video data with VActive
6	GPStopPolarity	✓	✓	0	0 = Active low at pin 1 = Active high at pin
7..8	Reserved	✓	✗	0x7	
9	HRefPolarityA	✓	✓	0	0 = Active low 1 = Active high
10	VRefPolarityA	✓	✓	0	0 = Active low 1 = Active high
11	VActivePolarity A	✓	✓	0	0 = Active low 1 = Active high
12	UseFieldA	✓	✓	0	0 = Disabled 1 = Enabled
13	FieldPolarityA	✓	✓	0	0 = Active low 1 = Active high
14	FieldEdgeA	✓	✓	0	0 = Inactive edge 1 = Active edge
15	VActiveVBIA	✓	✓	0	0 = Ignore VActive for VBI data 1 = Gate VBI data with VActive
16	InterlaceA	✓	✓	0	0 = Video is not interlaced 1 = Video is interlaced
17	ReverseDataA	✓	✓	0	0 = Disabled 1 = Enabled
18	HRefPolarityB	✓	✓	0	0 = Active low 1 = Active high
19	VRefPolarityB	✓	✓	0	0 = Active low 1 = Active high
20	VActivePolarity B	✓	✓	0	0 = Active low 1 = Active high
21	UseFieldB	✓	✓	0	0 = Disabled 1 = Enabled
22	FieldPolarityB	✓	✓	0	0 = Active low 1 = Active high
23	FieldEdgeB	✓	✓	0	0 = Inactive edge 1 = Active edge

24	VActiveVBIB	✓	✓	0	0 = Ignore VActive for VBI data 1 = Gate VBI data with VActive
25	InterlaceB	✓	✓	0	0 = Video is not interlaced 1 = Video is interlaced
26	ColorSpaceB	✓	✓	0	0 = YUV 1 = RGB
27	ReverseDataB	✓	✓	0	0 = Disabled 1 = Enabled
28	DoubleEdgeB	✓	✓	0	0 = Disabled 1 = Enabled
29	CCIR656A	✓	✓	0	0 = Disabled 1 = Enabled
30	InvertDoubleEdgeB	✓	✓	0	0 = Disabled 1 = Enabled
31	Reserved	✓	×	0	

VSDMACommandBase

Name	Type	Offset	Format
VSDMACommandBase	Video stream Control *Control register*	0x5AC8	Integer

Bits	Name	Read	Write	Reset	Description
0..3	Reserved	✓	✗	X	
4..31	Address	✓	✓	0	

Notes:

VSDMACommandCount

Name	Type	Offset	Format
VSDMACommandCount	Video stream Control *Control register*	0x5AD0	Integer

Bits	Name	Read	Write	Reset	Description
0..31	Count	✓	✓	0	

Notes:

VSDMAMode

Name	Type	Offset	Format
VSDMAMode	Video stream Control *Control register*	0x5AC0	Bitfield

Bits	Name	Read	Write	Reset	Description
0..21	Reserved	✓	✗	0	
22	Active	✓	✓	0	0 = DMA complete 1 = DMA running
23	MemType	✓	✓	0	0 = PCI 1 = AGP
24..25	Burst	✓	✓	0	Log2 of burst length
26	Reserved	✓	✗	0	
27	Align	✓	✓	0	0 = Disable 1 = Enable
28..31	Reserved	✓	✗	0	

Notes:

VSSerialBusControl

Name	Type	Offset	Format
VSSerialBusControl	Video stream Control	0x5810	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	DataIn	✓	✗	X	0 = Data line is low 1 = Data line is high
1	ClkIn	✓	✗	X	0 = Clock line is low 1 = Clock line is high
2	DataOut	✓	✓	1	0 = Drive data line low 1 = Tri-state data line
3	ClkOut	✓	✓	1	0 = Drive Clock line low 1 = Tri-state clock line
4	LatchedData	✓	✗	0	0 = Data latched at 0 1 = Data latched at 1
5	DataValid	✓	✓	0	0 = DataIn not valid 1 = DataIn valid
6	Start	✓	✓	0	0 = Has not passed through start state 1 = Has passed through start state
7	Stop	✓	✓	0	0 = Has not passed through stop state 1 = Has passed through stop state
8	Wait	✓	✓	0	0 = Do not insert wait states 1 = Insert wait states
9..31	Reserved	✓	✗	0	

Notes: Some bits in this register are set during operation and cleared by writing to the register with those bits set. The bits are DataValid, Start and Stop.

VSStatus

Name	Type	Offset	Format
VSStatus	Video stream Control *Control register*	0x5808	Bitfield

Bits	Name	Read	Write	Reset	Description
0	GPBusTimeOut	✓	✓	0	cleared by writing 1
1..7	Reserved	✓	×	0	
8	FifoOverflowA	✓	✓	0	cleared by writing 1
9	FieldOne0A	✓	×	0	
10	FieldOne1A	✓	×	0	
11	FieldOne2A	✓	×	0	
12	InvalidInterlaceA	✓	×	0	
13	BufferFieldA0	✓	×	0	
14	BufferFieldA1	✓	×	0	
15	BufferFieldA2	✓	×	0	
16	FifoUnderflowB	✓	✓	0	cleared by writing 1
17	FieldOne0B	✓	×	0	
18	FieldOne1B	✓	×	0	
19	FieldOne2B	✓	×	0	
20	InvalidInterlaceB	✓	×	0	
21	BufferFieldB0	✓	×	0	
22	BufferFieldB1	✓	×	0	
23	BufferFieldB2	✓	×	0	
24..31	Reserved	✓	×	0	

Notes:

VSAVideoStride SeeVSAVBIStride

4.9 Region 0 VGA Control (0x6000-0x6FFF)

The VGA registers generally follow industry VGA conventions. The registers described below are chip-specific varinats accessible both via VGA I/O and addressable memory (described here), togather with the index registers which support them (*GraphicsIndexReg* and *SequencerIndexReg*.). To read or write an indexed register first write the index value to the indexing register, then read/write the memory-mapped address (or VGA I/O Port).

4.9.1 Graphics Index Register

GraphicsIndexReg

Name	Type	Offset	Format
GraphicsIndexReg	VGA *Control register*	0x63CE	Bitfield

Bits	Name	Read	Write	Reset	Description
3:0	Index	✓	✓	X	This index points to one of the Graphics registers which will get read or written on the next I/O access to the GraphicsPort (0x3cf). The registers and their corresponding indices are: 0x0 SetResetReg 0x1 SetResetEnableReg 0x2 ColorCompareReg 0x3 DataRotateReg 0x4 ReadMapSelectReg 0x5 GraphicsModeReg 0x6 GraphicsMiscReg 0x7 ColorDontCareReg 0x8 BitMaskReg 0x9 Mode640Reg 0xa None : : 0xf None
7:4	Reserved	✓	×	0	Reserved

Notes: Writes to a register denoted 'None' have no effect as the write is simply discarded. Reading from a register denoted 'None' just returns zero.

Mode640Reg

Name	Type	Offset	Format
Mode640Reg	VGA *Control register*	0x63CF	Bitfield

Bits	Name	Read	Write	Reset	Description
2:0	BankA[2:0]	✓	✓	00	This field provides the additonal address bits needed when the horizontal screen resolution is 640 pixels and a host address is being made to the 64K region starting at address 0xa0000.
5:3	BankB[2:0]	✓	✓	00	This field provides the additional address bits needed when the horizontal screen resolution is 640 pixels and a host address is being made to the 64K region starting at address 0xb0000.
6	StartAddress16	✓	✓	00	The most significant bit of the StartAddress when mode 640 is enabled.
7	Enable	✓	✓	00	0 No action. 1 The VGA core operates in 640 resolution mode.

Notes: This register supports the 640 horizontal resolution modes used in SVGA. The BankA and BankB parts of this register are now obsolete. Programmers should use the sequencer registers BankALowReg, BankAHighReg, BankBLowReg, BankBHighReg instead. This register may be removed from future hardware

4.9.2 Sequencer Registers

SequencerIndexReg

Name	Type	Offset	Format
SequencerIndexReg	VGA *Control Register*	0x63C4	Bitfield

Bits	Name	Read	Write	Reset	Description
5:0	Index	✓	✓	X	This index points to one of the sequencer registers which will get read or written on the next I/O access to the SequencerPort (0x3c5). The registers and their corresponding indices are: 0x00 ResetReg 0x01 ClockModeReg 0x02 MapMaskReg 0x03 CharacterMapSelectReg 0x04 MemoryModeReg 0x05 VGAControlReg 0x06 LockExtended1Reg 0x07 LockExtended2Reg 0x08 BankALowReg 0x09 BankAHighReg 0x0a BankBLowReg 0x0b BankBHighReg 0x0c PCIControlReg 0x0d HLockShiftReg 0x0e VLockShiftReg 0x0f GenLockControlReg 0x10 .. 0x1f ScratchRegs 0x20 .. 0x23 IndirectBaseRegs 0x27 .. 0x3f None
7:6	Reserved	✓	×	0	Reserved

Notes:
- This register indexes data for the memory mapped *VGAControlReg* register and others shown below. To write to VGAControlReg first write a 0x05 to this regiater, then write data to VGAControlReg
- Writes to a register denoted 'None' have no effect as the write is simply discarded. Reading from a register denoted 'None' just returns zero.

4.9.2.1 Sequenced Registers

BankAHighReg

Name	Type	Offset	Format
BankAHighReg	VGA	0x635C index 0x09	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0,1	BankA9_8	✓	✓		This field holds the 2 high order bits of the 10-bit BankA base address. The 8 low order bits can be found in the BankALowReg. The BankA base address is used for bank switching the 0xa0000 region through the bypass (if enabled). The BankA bits provide the HBankA signals to the PCI interface.
2..7	Reserved	✓	✗	0	

Notes: To read/write this register, first write 0x0F to *SequencerIndexReg*. Not to be confused with Mode640Reg.BankA, which will become obsolete

BankALowReg

Name	Type	Offset	Format
BankALowReg	VGA	0x635C index 0x08	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	BankA7_0	✓	✓		This field holds the 8 low order bits of the 10-bit BankA base address. The 2 high order bits can be found in the BankAHighReg. The BankA base address is used for bank switching the 0xa0000 region through the bypass (if enabled). The BankA bits provide the HBankA signals to the PCI interface.

Notes: To read/write this register, first write 0x08 to *SequencerIndexReg*. Not to be confused with Mode640Reg.BankA, which will become obsolete.

BankBHighReg

Name	Type	Offset	Format
BankBHighReg	VGA	0x635C index 0x0B	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0,1	BankB9_8	✓	✓		This field holds the 2 high order bits of the 10-bit BankB base address. The 8 low order bits can be found in the BankBLowReg. The BankB base address is used for bank switching the 0xb0000 region through the bypass (if enabled). The BankB bits provide the HBankB signals to the PCI interface.
2…7	Reserved	✓	×	0	

Notes: To read/write this register, first write 0x0B to *SequencerIndexReg*

BankBLowReg

Name	Type	Offset	Format
VGAControlReg	VGA	0x635C index 0x0A	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	BankB7_0	✓	✓		This field holds the 8 low order bits of the 10-bit BankB base address. The 2 high order bits can be found in the BankBHighReg. The BankB base address is used for bank switching the 0xb0000 region through the bypass (if enabled). The BankB bits provide the HBankB signals to the PCI interface.

Notes: Not to be confused with Mode640Reg.BankB, which will become obsolete. To read/write this register, first write 0x0A to *SequencerIndexReg*

GenLockControlReg

Name	Type	Offset	Format
VGAControlReg	VGA	0x635C index 0x0F	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	Enable	✓	✓		If set, allows the VTG to be synchronized to an external video source. This causes the horizontal & vertical sync starts & blank ends to be delayed. Sync starts are delayed until the arrival of the ExtHSync & ExtVSync signals. Blank ends are delayed by the numbers specified in the HLockShiftReg & VLockShiftReg registers.
1…7	Reserved	✓	×	0	

Notes: This register is not supported in current releases. Use software Genlock where necessary.

HLockShiftReg

Name	Type	Offset	Format
HLockShiftReg	VGA	0x635C index 0x0D	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7		✓	✓		If genlocking is enabled, this field specifies the number of characters by which the horizontal blank end is delayed.

Notes: This register is not supported in current releases – use software genlock where required.

IndirectBaseReg[0x0…0x3]

Name	Type	Offset	Format
IndirectBaseReg[0x0…0x3	VGA	0x635C	Bitfield
		index 0x20 – 0x23	
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7		✓	✗	x	These 4 registers follow the state of the HIndirectBase signals from the PCI interface. IndirectBaseReg[0] returns bits 7..0, IndirectBaseReg[1] returns bits 15..8, IndirectBaseReg[2] returns bits 23..16, and IndirectBaseReg[3] returns bits 31..24.

Notes: To read from this register, first write the index value (0x20 to 0x23) to *SequencerIndexReg*, then read the required index entries.

LockExtended1Reg

Name	Type	Offset	Format
LockExtended1Reg	VGA	0x63C5	Bitfield
		index 0x06	
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Lock	✗	✓		These 2 registers act as a lock for the extended registers. On reset extended registers are locked – they cannot be written and read back as 0, and the sequencer index behaves as a 3-bit index. Writing the value 0x3d to *LockExtended1Reg* followed by 0xdb to *LockExtended2Reg* unlocks the extended registers. Writing any other values locks them.
8…31	Reserved	✓	✗	0	

Notes: To read/write this register, first write 0x06 to *SequencerIndexReg*.

LockExtended2Reg

Name	**Type**	**Offset**	**Format**
LockExtended2Reg	VGA	0x63C5 index 0x07	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Lock	✗	✓		Acts as a lock for the extended registers. On reset extended registers are locked - they cannot be written and read back as 0, and the sequencer index behaves as a 3-bit index. Writing the value 0x3d to LockExtended1Reg followed by 0xdb to LockExtended2Reg unlocks the extended registers. Writing any other values locks them.

Notes: To read/write this register, first write 0x07 to *SequencerIndexReg*.

PCIControlReg

Name	Type	Offset	Format
PCIControlReg	VGA	0x635C index 0x0C	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	BankEnable	✓	✓		If set, enables bank switching of the 0xa0000/0xb0000 regions through the bypass, using the 10-bit BankA/BankB base addresses. This bit provides the HBankEnable signal to the PCI interface.
1	IndirectEnable	✓	✓		If set, enables access to chip registers via I/O ports 0x3b0/0x3b1/0x3d0/0x3d1. This bit provides the HIndirectEnable signal to the PCI interface.
2…7	Reserved	✓	✗	0	Reserved.

Notes: To read/write this register, first write 0x0C to *SequencerIndexReg*

ScratchReg[0x0…0xf]

Name	Type	Offset	Format
ScratchReg[0x0…0xF]	VGA	0x635C index 0x10 to 0x1F	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7		✓	✓		These registers are available for use as an information store and do not affect the VGA operation.

Notes: To read/write this register first write the index value (0x10 to 0xF) to *SequencerIndexReg*, then read the required index entries.

VGAControlReg

Name	Type	Offset	Format
VGAControlReg	VGA	0x63C5 index 0x05	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	EnableHost MemoryAccess	✓	✓		Controls access to the display memory by the host. 0 No access to the display memory is made in response to host VGA memory accesses. Writes are ignored and reads always return zero. All the host bus cycles are completed as normal. 1 Normal access to the display memory occurs. This bit is further qualified by the VGAEnable signal which acts as a global disable.
1	EnableHost DacAccess	✓	✓		Controls access to the RAMDAC by the host. 0 No access to the RAMDAC is made in response to host Dac accesses. Writes are ignored and reads always return zero. All the host bus cycles are completed as normal. 1 Normal access to the RAMDAC occurs. This bit is further qualified by the VGAEnable signal which acts as a global disable.

2	Enable Interrupts	✓	✓		0 Prevents any interrupts from being generated by the VGA core. 1 Enables interrupt generation from the VGA core providing the VerticalSyncEndReg.DisableVerticalInterrupt field is set to zero. This bit is further qualified by the VGAEnable signal which acts as a global disable. This additional enable bit is provided so the VGA core can be disabled from one place.
3	EnableVGA Display	✓	✓		Controls access to the display memory by the Memory Reader for the purpose of keeping the display refreshed. It also tells (on the VGAVidSelect signal) the video select logic external to the VGA core that the display should be driven from the VGA core. 0 No accesses to display memory are to be made and the video source should not be the VGA core. The Memory Reader, Attribute Controller and Video Timing Generator are held in their reset state. 1 Accesses to the display memory are made and the video to be displayed comes from the VGA core. This bit is further qualified by the VGAEnable signal which acts as a global disable.
4	DacAddr2	✓	✓		This bit extends the RAMDAC address range.
5	DacAddr3	✓	✓		This bit extends the RAMDAC address range.
6	EnableVTG	✓	✓	x	0 Stops the VTG running and producing sync pulses. 1 Enables the VTG to run and produce sync pulses. This bit only has an effect when the VGA display has been disabled by EnableVGADisplay. When the display has been disabled by VGAEnable this bit is ignored. When the VGA dispaly is active then this bit is ignored.
7	InvertVBlank	✓	✓	0	0 No Invert VBlank. 1 Invert VBlank

Notes:

- On reset EnableHostMemoryAccess, EnableHostDacAccess and EnableVGADisplay are enabled, EnableInterrupts is disabled and DacAddr2 and DacAddr3 bits are set to 0, InvertVBlank is set to 0.
- This is a non standard VGA register.
- To read/write this register, first write 0x05 to *SequencerIndexReg*

VLockShiftReg

Name	Type	Offset	Format
VLockShiftReg	VGA	0x635C index 0x0E	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7		✓	✓	0	If genlocking is enabled, this field specifies the number of scanlines by which the vertical blank end is delayed.

Notes: This register is not supported in current releases.

4.10 Region 0 Texture Data FIFO (0x7000-0x7FFF)

No 0x7000 series registers are listed.

4.11 Region 3 Indirect Addressing

IndirectAccess

Name	Type	Offset	Format
IndirectAccess	Region 3 *Control register*	0x0C	Integer

Bits	Name	Read	Write	Reset	Description
0..31	Reserved	×	×	0	Accessing any part of these 32 bits triggers an indirect access to the location addressed by IndirectAddr. A write here will trigger the write of IndirectData into the location. A read here will trigger the read of the location into IndirectData. The access is further masked by the byte enables specified in Indirect ByteEn.

Notes:

IndirectAddr

Name	Type	Offset	Format
IndirectAddr	Region 3 *Control register*	0x08	Integer

Bits	Name	Read	Write	Reset	Description
0..28	Offset	✓	✓	0	These bits specify the offset of the location to be accessed.
29..31	Region	✓	✓	0	These bits specify the region of the location to be accessed. If region is 1, accesses are to region 1. If region is 2, accesses are to region 2. If region is 3, accesses are to region 3. If region is 4, accesses are to region 4. Otherwise accesses are to region 0.

Notes:

IndirectByteEnable

Name	**Type**	**Offset**	**Format**
IndirectByteEnable	Region 3 *Control register*	0x00	Integer

Bits	Name	Read	Write	Reset	Description
0..3	Byte Enables	✓	✓	0	These four bits specify the mask to apply to accesses to the location by IndirectAddr. bit 0 set to 1 enables IndirectData byte 0 bit 1 set to 1 enables IndirectData byte 1 bit 2 set to 1 enables IndirectData byte 2 bit 3 set to 1 enables IndirectData byte 3
4..31	Reserved	✓	✗	0	

Notes:

IndirectData

Name	**Type**	**Offset**	**Format**
IndirectData	Region 3 *Control register*	0x04	Integer

Bits	Name	Read	Write	Reset	Description
0..31	Data	✓	✓	0	These 32 bits hold the data to be written to, or read from, the location addressed by IndirectAddr. The access is further masked by the byte enables specified in IndirectByteEn.

Notes:

5

Graphics Registers

This chapter lists PERMEDIA 3 graphics core ('software') registers in region 0, offset group 0x8000-0xFFFF. Within this group the registers are listed alphanumerically. All other registers are described in chapter 4. Global cross-reference listings in alphanumeric and offset order are available in chapter 6.

Register details have the following format information:

Name	The register's name.
Type	The region in which the register functions.
Offset	The offset of this register from the base address of the region.
Format	Can be bitfield or integer.
Bit	Bit Name
Read	Indicates whether the register bit can be read from. A ✓ mark indicates the register can be read from, a ✕ indicates the register bit is not readable.
Write	Indicates whether the register bit can be written to. A ✓ mark indicates the register can be written to, a ✕ indicates the register bit is not writable.
Reset	The value of the register following hardware reset.
Description	In the register descriptions:
Reserved	Indicates bits that may be used in future members of the PERMEDIA family. To ensure upwards compatibility, any software should not assume a value for these bits when read, and should always write them as zeros.
Not Used / Unused	Indicates bits that are adjacent to numeric fields. These may be used in future members of the PERMEDIA family, but only to extend the dynamic range of these fields. The data returned from a read of these bits is undefined. When a Not Used field resides in the most significant position, a good convention to follow is to sign extend the numeric value, rather than masking the field to zero before writing the register. This will ensure compatibility if the dynamic range is increased in future members of the PERMEDIA family.

For enumeration fields that do not specify the full range of possible values, only the specified values should be used. An example of an enumeration field is the comparison field in the DepthMode register. Future members of the PERMEDIA family may define a meaning for the unused values.

AlphaBlendAlphaMode
AlphaBlendAlphaModeAnd
AlphaBlendAlphaModeOr

Name	**Type**	**Offset**	**Format**
AlphaBlendAlphaMode	Alpha Blend	0x AFA8	Bitfield
AlphaBlendAlphaModeAnd	Alpha Blend	0x AD30	Bitfield Logic Mask
AlphaBlendAlphaModeOr	Alpha Blend	0x AD38	Bitfield Logic Mask
	Control registers		

Bits	Name	Read[1]	Write	Reset	Description
0	Enable	✓	✓	x	When set causes the fragment's alpha to be alpha blended under control of the remaining bits in this register. When clear the fragment alpha remains unchanged (but may later to affected by the chroma test).
1…4	SourceBlend	✓	✓	x	This field defines the source blend function to use. See the table below for the possible options.
5…7	DestBlend	✓	✓	x	This field defines the destination blend function to use. See the earlier table for the possible options.
8	Source TimesTwo	✓	✓	x	This bit, when set causes the source blend result to be multiplied by two before it is combined with the dest blend result. When this bit is clear no multiply occurs.
9	DestTimes Two	✓	✓	x	This bit, when set causes the dest blend result to be multiplied by two before it is combined with the source blend result. When this bit is clear no multiply occurs.
10	Invert Source	✓	✓	x	This bit, when set, causes the incomming source data to be inverted before any blend operation takes place.
11	Invert Dest	✓	✓	x	This bit, when set, causes the incomming dest data to be inverted before any blend operation takes place.
12	NoAlpha Buffer	✓	✓	x	When this bit is set the source alpha value is always set to 1.0. This is typically used when no retained alpha buffer is present but will also override any retained alpha value if one is present. Color formats with no alpha field defined automatically have their alpha value set to 1.0 regardless of the state of this bit.
13	Alpha Type	✓	✓	x	This bit selects which set of equations are to be used for the alpha channel. 0 = OpenGL 1 = Apple

[1] Logic Op register readback is via the main register.

14	Alpha Conversion	✓	✓	x	This bit selects how alpha component less than 8 bits wide are converted to 8 bit wide values prior to the alpha blend calculations. The options are 0 = Scale 1 = Shift
15	Constant Source	✓	✓	x	This bit, when set, forces the Source color to come from the AlphaSourceColor register (in 8888 format) instead of the framebuffer. 0 = Use framebuffer alpha 1 = Use AlphaSourceColor register alpha value.
16	Constant Dest	✓	✓	x	This bit, when set, forces the destination color to come from the AlphaDestColor register (in 8888 format) instead of the fragment's color. 0 = Use fragment's alpha. 1 = Use AlphaDestColor register alpha value
17…19	Operation	✓	✓	x	This field selects how the source and destination blend results are to be combined. The options are: 0 = Add 1 = Subtract (i.e. S - D) 2 = Subtract reversed (i.e. D - S) 3 = Minimum 4 = Maximum

Notes The Alpha Conversion bit selects the conversion method for alpha values read from the framebuffer.

- The Scale method linearly scales the alpha values to fill the full range of an 8 bit value. This method is preferable when, for example, downloading an image with fewer bits per pixel into a deeper (i.e. more bits per pixel) framebuffer.
- The Shift method just left shifts by the appropriate amount to make the component 8 bits wide. This method is preferable when blending into a dithered framebuffer as it preserves the framebuffer alpha when fragment alpha does not contribute to it.

Alpha is controlled separately from color to allow, for example, the situation in antialiasing where it represents coverage - this must be linearly scaled to preserve the 100% covered state.

The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

The table below shows the different color modes supported. In the R, G, B and A columns the nomenclature n@m means this component is n bits wide and starts at bit position m in the framebuffer. The least significant bit position is 0 and a dash in a column indicates that this component does not exist for this mode.

In the case of the RGB formats where no Alpha is shown then the alpha field is set to 255. In this case the NoAlphaBuffer bit in the AlphaBlendAlphaMode register should be set which causes the alpha component to be set to 255.

Two color ordering formats are supported, namely ABGR and ARGB, with the right most letter representing the color in the least significant part of the word. This is controlled by the Color Order bit in the *AlphaBlendColorMode* register, and is easily implemented by just swapping the R and B components after conversion into the internal format. The only exception to this are the 3:3:2 formats where the actual bit fields extracted from the framebuffer data need to be modified as well because the R and B components are differing widths. CI processing is not affected by this and the result is always on internal R channel.

The format to use is held in the *AlphaBlendColorMode* register. Note that in OpenGL alpha blending is not defined for CI mode..

When converting a Color Index value to the internal format any unused bits are set to zero

	Format	Color Order	Name	Internal Color Channels			
				R	G	B	A
Colour	0	BGR	8:8:8:8	8@0	8@8	8@16	8@24
	1	BGR	4:4:4:4	4@0	4@4	4@8	4@12
	2	BGR	5:5:5:1	5@0	5@5	5@10	1@15
	3	BGR	5:6:5	5@0	6@5	5@11	-
	4	BGR	3:3:2	3@0	3@3	2@6	-
	0	RGB	8:8:8:8	8@16	8@8	8@0	8@24
	1	RGB	4:4:4:4	4@8	4@4	4@0	4@12
	2	RGB	5:5:5:1	5@10	5@5	5@0	1@15
	3	RGB	5:6:5	5@11	6@5	5@0	-
	4	RGB	3:3:2	3@5	3@2	2@0	-
CI	15	X	CI8	8@0	0	0	0

AlphaBlendColorMode
AlphaBlendColorModeAnd
AlphaBlendColorModeOr

Name	Type	Offset	Format
AlphaBlendColorMode	Alpha Blend	0x AFA0	Bitfield
AlphaBlendColorModeAnd	Alpha Blend	0x ACB0	Bitfield Logic Mask
AlphaBlendColorModeOr	Alpha Blend	0x ACB8	Bitfield Logic Mask
	Control registers		

Bits	Name	Read[2]	Write	Reset	Description
0	Enable	✓	✓	x	When set causes the fragment's color to be alpha blended under control of the remaining bits in this register. When clear the fragment color remains unchanged (but may later to effected by the chroma test).
1…4	SourceBlend	✓	✓	x	This field defines the source blend function to use. See the table in the *AlphaBlendColorMode* register for the possible options
5…7	DestBlend	✓	✓	x	This field defines the destination blend function to use. See the table in the *AlphaBlendColorMode* register for the possible options
8	Source TimesTwo	✓	✓	x	This bit, when set causes the source blend result to be multiplied by two before it is combined with the dest blend result. When this bit is clear no multiply occurs
9	DestTimes Two	✓	✓	x	This bit, when set causes the dest blend result to be multiplied by two before it is combined with the source blend result. When this bit is clear no multiply occurs
10	InvertSource	✓	✓	x	This bit, when set, causes the incomming source data to be inverted before any blend operation takes place
11	InvertDest	✓	✓	x	This bit, when set, causes the incomming dest data to be inverted before any blend operation takes place
12…15	Color Format	✓	✓	x	This field defines framebuffer color formats. See the table in the *AlphaBlendColorMode* register for the possible options
16	ColorOrder	✓	✓	x	This bit selects the color order in the framebuffer: 0 = BGR 1 = RGB
17	Color Conversion	✓	✓	x	This bit selects how color components less than 8 bits wide are converted to 8 bit wide values prior to the alpha blend calculations. The options are 0 = Scale 1 = Shift

[2] Logic Op register readback is via the main register

18	Constant Source	✓	✓	x	This bit, when set, forces the Source color to come from the *AlphaSourceColor* register (in 8888 format) instead of the framebuffer. 0 = Use framebuffer 1 = Use AlphaSourceColor register
19	ConstantDest	✓	✓	x	This bit, when set, forces the destination color to come from the *AlphaDestColor* register (in 8888 format) instead of the fragment's color. 0 = Use fragment's color. 1 = Use *AlphaDestColor* register.
20…23	Operation	✓	✓	x	This field selects how the source and destination blend results are to be combined. The options are: 0 Add 1 Subtract (i.e. S - D) 2 Subtract reversed (i.e. D - S) 3 Minimum 4 Maximum
24	SwapSD	✓	✓	x	This bit, when set causes the source and destination pixel values to be swapped. The main use for this is to allow a downloaded color value to be in a format other than 8888 and use this unit to do color conversion.

Notes *AlphaBlendColor* combines the fragment's Color with the Color stored in the framebuffer using the alpha blend equations, to create lighting or translucncy effects for example. Alpha blending only works for pixels stored in the RGBA format (since Alpha values are not specified in color-index mode). After blending is done the new blended Color replaces the former Color. If alpha blending is disabled then the Color field passes the alpha blend unchanged.

The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

AlphaDestColor

Name	**Type**	**Offset**	**Format**
AlphaDestColor	Alpha Blend *Control register*	0xAF88	Bitfield

Bits	Name	Read	Write	Reset	Description
0..7	R	✓	✓	x	Red
8..15	G	✓	✓	x	Green
16..23	B	✓	✓	x	Blue
24..31	A	✓	✓	x	Alpha

Notes: This register holds the destination color to use instead of the fragment color when ConstantDest (in *AlphaBlendcolorMode* or *AlphaBlendAlphaMode*) is enabled. Each color component has a separate boundary held as an unsigned 8-bit number from Red (least significant bit) to Alpha.

AlphaSourceColor

Name	**Type**	**Offset**	**Format**
AlphaSourceColor	Alpha Blend *Control register*	0xAF80	Integer

Bits	Name	Read	Write	Reset	Description
0..7	R	✓	✓	x	Red
8..15	G	✓	✓	x	Green
16..23	B	✓	✓	x	Blue
24..31	A	✓	✓	x	Alpha

Notes: This register holds the source color to use instead of the framebuffer color when ConstantSource (in AlphaBlendcolorMode or AlphaBlendAlphaMode) is enabled. Each color component has a separate boundary held as an unsigned 8-bit number from Red (least significant bit) to Alpha.

AlphaTestMode
AlphaTestModeAnd
AlphaTestModeOr

Name	**Type**	**Offset**	**Format**
AlphaTestMode	AlphaBlend	0x 8800	Bitfield
AlphaTestModeAnd	AlphaBlend	0x ABF0	Bitfield Logic Mask
AlphaTestModeOr	AlphaBlend	0x ABF8	Bitfield Logic Mask
	Control registers		

Bits	Name	Read[3]	Write	Reset	Description
0	Enable	✓	✓	x	When set causes the fragment's alpha value to be tested under control of the remaining bits in this register. If the alpha test fails then the fragment is discarded. When this bit is clear the fragment alway passes the alpha test. 0 = Disable 1 = Enable
1…3	Compare	✓	✓	x	This field defines the unsigned comparison function to use. The options are: 0 = Never 1 = Less 2 = Equal 3 = Less Equal 4 = Greater 5 = Not Equal 6 = Greater Equal 7 = Always The comparison order is as follows: result = fragment, Alpha Compare Function, reference, Alpha.
4…11	Reference	✓	✓	x	This field holds the 8 bit reference alpha value used in the comparison.
12…31	Unused	0	0	x	

Notes The Alpha Test, if enabled, compares the alpha value of a fragment, after coverage weighting, against a reference value and if the compare passes the fragment is allowed to continue. If the comparison fails the fragment is culled and will not be drawn.

[3] Logic Op register readback is via the main register

AntialiasMode
AntialiasModeAnd
AntialiasModeOr

Name	**Type**	**Offset**	**Format**
AntialiasMode	Alpha Test	0x 8808	Bitfield
AntialiasModeAnd	Alpha Test	0x ABF0	Bitfield Logic Mask
AntialiasModeOr	Alpha Test	0x ABF8	Bitfield Logic Mask
	Control registers		

Bits	Name	Read[4]	Write	Reset	Description
0	Enable	✓	✓	x	When set causes the fragment's alpha value to be scaled under control of the remaining bits in this register and the coverage value. When this bit is clear the fragment's alpha value is not changed. 0 = Disable 1 = Enable
1	Color Mode	✓	✓	x	This bit defines the color format the fragment's color is in: 0 = RGBA 1 = CI
2	Scale Color	✓	✓	x	This bit, when set allows the coverage value to scale the RGB components as well as the alpha component. When this bit is reset only the alpha component is scaled. This allows antialiasing of pre multiplied images used in compositing.
3…31	Unused	0	0	x	

Notes: The register controls the operation of antialiasing. When the unit is enabled:

- In Color Index (CI) mode the bottom 4 bits of the color index of a fragment is replaced by the coverage value scaled by 15/256, where the result is rounded to the nearest integer.
- In RGBA mode the alpha component of a fragment is multiplied by the coverage value, but the RGB components are not changed unless ScaleColor is also enabled

When antialiased primitives are being rendered the fragment's color is weighted by the percentage area of the pixel the fragment covers. An approximation to the area covered is calculated.

If antialiasing is disabled then the color is passed onto the alpha test stage unchanged. Note that the CoverageEnable bit in the *Render* command must also be set to enable antialiasing.

The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

[4] Logic Op register readback is via the main register only

AreaStippleMode
AreaStippleModeAnd
AreaStippleModeOr

Name	Type	Offset	Format
AreaStippleMode	Stipple	0x81A0	Bitfield
AreaStippleModeAnd	Stipple	0xABD0	Bitfield Logic Mask
AreaStippleModeOr	Stipple	0xABD8	Bitfield Logic Mask
	Control registers		

Bits	Name	Read[5]	Write	Reset	Description
0	Enable	✓	✓	x	This field, when set, enables area stippling. The AreaStippleEnable bit in *Render* must also be set for this to have an effect.
1..3	X address select:	✓	✓	x	0 = 1 bit 1 = 2 bit 2 = 3 bit 3 = 4 bit 4 = 5 bit
4..6	Y address select:	✓	✓	x	0 = 1 bit 1 = 2 bit 2 = 3 bit 3 = 4 bit 4 = 5 bit
7..11	X Offset	✓	✓	x	This field holds the offset to add to the X value before it is used to index into the stipple bit. This allows a window relative stipple pattern to be selected when the coordinates are given in screen relative format.
12..16	Y Offset	✓	✓	x	This field holds the offset to add to the Y value before it is used to index into the area stipple pattern table. This allows a window relative stipple pattern to be selected when the coordinates are given in screen relative format.
17	Invert Stipple Pattern	✓	✓	x	0 = No Invert 1 = Invert
18	Mirror X	✓	✓	x	0 = No Mirror 1 = Mirror
19	Mirror Y	✓	✓	x	0 = No Mirror 1 = Mirror
20	OpaqueSpan	✓	✓	x	This bit, when set, allows the area stipple pattern to modify the color mask, otherwise the pixel mask is modified.
21…25	XTableOffset	✓	✓	x	This field allows a sub area stipple pattern to be extracted from the area stipple table, i.e. the area stipple table is treated as a cache of smaller stipple patterns.
26…30	YTableOffset	✓	✓	x	This field allows a sub area stipple pattern to be extracted from the area stipple table, i.e. the area stipple table is treated as a cache of smaller stipple patterns.
31	Unused	0	0	x	

[5] Logic Op register readback is via the main register only

Notes:	1.	This register controls Area Stippling. This involves applying the correct stipple pattern (mask) which can also be mirrored or inverted. The least significant bits of the fragment's XY coordinates index into a 2D stipple pattern. If the selected bit is set the fragment passes the test, otherwise it fails. An offset is added to the XY coordinate and the result optionally mirrored and/or inverted before the stipple bit is accessed.
	2.	Both the AreaStippleEnable bit in the *Render* command and the enable in the *AreaStippleMode* register must be set, to enable the area stipple test.
	3.	The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

AreaStipplePattern [0…15]
AreaStipplePattern [16…31]

Name	**Type**	**Offset**	**Format**
AreaStipplePattern	Stipple *Control register*	0x8200 – 82F8	Bitmask

Bits	Name	Read	Write	Reset	Description
0…31	Mask	✓	✓	x	32 bit mask for area pattern data

Notes: These 32 registers provide the bitmask which enables and disables corresponding fragments for drawing when rasterizing a primtive with area stippling. They hold the LSBs and MSBs of area pattern data. The Y' value in the StippleMode register selects the row in the stipple RAM (row zero is at AreaStipplePattern[0]) and this is the first value of the AreaStippleMask.

AStart

Name	**Type**	**Offset**	**Format**
AStart	Color *Control register*	0x87C8	Fixed point number

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	Unused	0	0	x	

Notes: Used to set the initial Alpha value of a vertex when in Gouraud shading mode. The format is 24 bit 2's complement fixed point numbers in 9.15 format.

BackgroundColor

Name	Type	Offset	Format
BackgroundColor	Logic Ops *Control register*	0xB0C8	Integer

Bits	Name	Read	Write	Reset	Description
0...31	Background Color	✓	✓	x	32 bit integer

Notes: With ForegroundColor, holds the foreground and background color values. A background pixel is a pixel whose corresponding bit in the color mask is zero. The color format is in the raw framebuffer format and 8 or 16 bit pixels are automatically replicated to fill the 32 bits of register.

BasePageOfWorkingSet

Name	Type	Offset	Format
BasePageOfWorkingSet	Texture Read *Control register*	0xB4C8	Integer

Bits	Name	Read	Write	Reset	Description
0…15	Page number	✓	✓	x	16 bit integer value from 0 to 65535
15…31	Unused	0	0	x	

Notes: Holds the page number of the start of the region of memory to be used as the working set. This is measured in units of 4K bytes from 0 (the first byte address with respect to P3's view of the memory map). This allows the Physical Page Allocation Table to be smaller as it doesn't have to include low memory locations reserved for Z buffer, color buffers, etc. The legal range of values is 0…65535.

Before any logical or virtual texture management can be done there are a number of areas which need to be initialised (in addition to the usual mode, etc. register initialisation):

- Space for the Logical Texture Page Table must be reserved in the local buffer and the table initialised to zero. The LogicalTexturePageAddr and *LogicalTexturePageTableLength* must be set up.
- Space for the working set must be reserved in the local buffer and/or framebuffer. This need not be physically consecutive pages. The *BasePageOfWorkingSet* register is set up.

BasePageOfWorkingSetHost

Name	Type	Offset	Format
BasePageOfWorkingSet Host	Texture Read	0xB4E0	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…19	Page number	✓	✓	x	20 bit integer value.

Notes: This 20 bit register holds the page number of the start of the region of host memory to be used as the working set. This is a 256MByte region and can be positioned anywhere in the 4GByte host address range. This is measured in units of 4K bytes from 0 (the first byte address in the physical memory map).

BitMaskPattern

Name	Type	Offset	Format
BitMaskPattern	Rasterizer	0x8068	Integer
	Command and Control register		

Bits	Name	Read	Write	Reset	Description
0..31	Bitmask	✓	✓	x	32 bit value

Notes: Value used to control the bit mask stipple operation (if enabled). Fragments are accepted or rejected based on the current BitMask test modes defined by the RasterizerMode register. Note: the SyncOnBitmask bit in the Render command must also be enabled.

The bit mask is written in the BitMaskPattern register and can be modified in a number of ways before being used. These modifications are applied in the order below and are enabled using the corresponding bit in the RasterizerMode register.

As each pixel in the primitive is generated one bit of the bit mask is consumed. Internally the bits are always consumed from the least significant end towards the most significant end, however the MirrorBitMask effectively reverses this order.

BitMaskPattern Application Bits in the RasterizerMode Register		
Mode	**Raster-izer Mode Bit no.**	**Description (See *RasterizerMode* register for details)**
ByteSwapBitMask	7,8	Byte swaps the bit mask pattern as directed by the *BitMaskByteSwapMode*. This allows the bitmasks used internally for Windows or WindowsNT to be used directly
MirrorBitMask	0	The bit mask pattern is mirrored so bit 0 become bit 31, bit 1 becomes bit 30, etc. Bit 0 is the least significant bit. This feature allows the left most pixel in a window to be assigned to the most or least significant bit in the bit mask pattern.
InvertBitMask	1	The bit mask pattern is inverted before it is used so that fragments associated with '0' bits are now written instead of fragments associated with '1' bits. The inversion is useful when two passes are needed to draw the primitive, for example to draw the foreground pixels using a different logical operation to the background pixels for a character.
BitMaskPacking	9	Selects whether the bit mask pattern is packed so that adjacent rows butt together to minimise the number of words to transfer for the whole pattern. If not then a new bit mask pattern is required for every scanline. For span fills a new bit mask pattern *must* be provided at the start of every scanline.
BitMaskOffset	10..14	Determines the first bit to use in the bit mask pattern for the first bit mask pattern on a scanline. Subsequent bit masks will always start at bit 0 until the next scanline is encountered. The default is zero and the bit position refers to the position *after* any byte swapping or mirroring has been done. This allows the source and destination rectangle alignments to be different.

BorderColor0
BorderColor1

Name	**Type**	**Offset**	**Format**
BorderColor0	Texture	0x84A8	Bitfield
BorderColor1	Texture	0x84F8	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	R	✓	✓	x	Red
8…15	G	✓	✓	x	Green
16…23	B	✓	✓	x	Blue
24…31	A	✓	✓	x	Alpha

Notes: If a border has not been provided in the texture map, but a border texel is needed, they are taken from the BorderColor registers. BorderColor0 holds the border color to be used for Texels T0…T3. Its format is red in byte 0, green in byte 1, blue in byte 2 and alpha in byte 3. BorderColor1 holds the border color to be used for Texels T4…T7. Its format is identical.

BStart

Name	**Type**	**Offset**	**Format**
BStart	Color	0x87B0	Fixed point number
	Control register		

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	Unused	0	0	x	

Notes: Used to set the initial Blue value for a vertex when in Gouraud shading mode. The value is 24 bit 2's complement fixed point numbers in 9.15 format.

ChromaFailColor

Name	Type	Offset	Format
ChromaFailColor	Color *Control register*	0xAF98	Bitfield

Bits	Name	Read	Write	Reset	Description
0..7	R	✓	✓	x	Red
8..15	G	✓	✓	x	Green
16..23	B	✓	✓	x	Blue
24..31	A	✓	✓	x	Alpha

Notes: This register holds the chroma color to use when the chroma test is enabled and the chroma operation is substitute fail color. Its format is 8 bit ABGR components packed into a 32 bit word with R in the LS byte.

ChromaLower

Name	Type	Offset	Format
ChromaLower	Color *Control register*	0x8F10	Bitfield

Bits	Name	Read	Write	Reset	Description
0..7	R	✓	✓	x	Red
8..15	G	✓	✓	x	Green
16..23	B	✓	✓	x	Blue
24..31	A	✓	✓	x	Alpha

Notes: This register holds the lower bound color for the chroma test. Each color component has a seperate boundary held as an unsigned 8 bit number with Red in the lower byte, then green, then blue and finaly in the upper byte alpha. The test is inclusive so the fragment is in range if all its components are less than or equal to the upper bound and greater than or equal to the lower bound. The options are to reject the fragment so nothing gets drawn or the color is replaced by the value held in the ChromaPassColor or ChromaFailColor registers. *Note this is different to GLINT MX*

ChromaPassColor

Name	Type	Offset	Format
ChromaPassColor	Color *Control register*	0xAF90	Bitfield

Bits	Name	Read	Write	Reset	Description
0..7	R	✓	✓	x	Red
8..15	G	✓	✓	x	Green
16..23	B	✓	✓	x	Blue
24..31	A	✓	✓	x	Alpha

Notes: This register holds the chroma color to use when the chroma test is enabled and the chroma operation is substitute pass color. Its format is 8 bit ABGR components packed into a 32 bit word with R in the LS byte.

ChromaTestMode
ChromaTestModeAnd
ChromaTestModeOr

Name	Type	Offset	Format
ChromaTestMode	Alpha Blend	0x8F18	Bitfield
ChromaTestModeAnd	Alpha Blend	0xACC0	Bitfield Logic Mask
ChromaTestModeOr	Alpha Blend	0xACC8	Bitfield Logic Mask
	Control registers		

Bits	Name	Read[6]	Write	Reset	Description
0	Enable	✓	✓	x	When set enables chroma testing under control of the remaining bits in this register. When clear no chroma test is done.
1…2	Source	✓	✓	x	This field selects which color (after any suitable conversion) is to be used for the chroma test. The values are: 0 = FBSourceData 1 = FBData 2 = Input Color (from fragment) 3 = Output Color (after any alpha blending)
3…4	PassAction	✓	✓	x	This field defines what action is to be taken if the chroma test passes (and is enabled). The options are: 0 = Pass 1 = Reject 2 = Substitute ChromaPassColor 3 = Substitute ChromaFailColor
5…6	FailAction	✓	✓	x	This field defines what action is to be taken if the chroma test fails (and is enabled). The options are: 0 = Pass 1 = Reject 2 = Substitute ChromaPassColor 3 = Substitute ChromaFailColor
7…31	Unused	0	0	x	

Notes: Used to test the fragment's color against a range of colors after alphablending. The chroma test is enabled by the enable bit (0) in the register. Note: incompatible with MX programming.

The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

[6] Logic Op register readback is via the main register only

ChromaUpper

Name	Type	Offset	Format
ChromaUpper	Color *Control register*	0x8F08	Bitfield

Bits	Name	Read	Write	Reset	Description
0..7	R	✓	✓	x	Red
8..15	G	✓	✓	x	Green
16..23	B	✓	✓	x	Blue
24..31	A	✓	✓	x	Alpha

Notes: This register holds the upper bound color for the chroma test. Each color compoent has a seperate boundary held as an unsigned 8 bit number with Red in the lower byte, then greeen, then blue and finaly in the upper byte alpha. The test is inclusive so the a fragment is in range if all its components are less than or equal to the upper bound and greater than or equal to the lower bound. The options are to reject the fragment so nothing gets drawn or the color is replaced by the value held in the ChromaPassColor or ChromaFailColor registers. *Note this is different to GLINT MX*

Color

Name	Type	Offset	Format
Color	Host In *Control register*	0x87F0	Bitfield

Bits	Name	Read	Write	Reset	Description
0...7	Red	✓	✓	x	
8...15	Green	✓	✓	x	
16...23	Blue	✓	✓	x	
24…31	Alpha	✓	✓	x	

Notes: This register is used in conjunction with the *SyncOnHost* bit in the **Render** command to trigger fragment generation under Host control.

ColorDDAMode
ColorDDAModeAnd
ColorDDAModeOr

Name	**Type**	**Offset**	**Format**
ColorDDAMode	Color	0x87E0	Bitfield
ColorDDAModeAnd	Color	0xABE0	Bitfield Logic Mask
ColorDDAModeOr	Color	0xABE8	Bitfield Logic Mask
	Control registers		

Bits	Name	Read[7]	Write	Reset	Description
1	Enable	✓	✓	x	This bit, when set, causes the current color to be generated.
2	Shading	✓	✓	x	Selects the shading mode. The two options are: 0 = Flat – the color is taken from the Constant Color register. 1 = Gouraud – the color is taken from the DDAs.
3…31	Unused	0	0	x	

Notes: The ColorDDAMode register controls the operation of the Color DDA unit using the Enable and Shading bits. The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

CommandInterrupt

Name	**Type**	**Offset**	**Format**
CommandInterrupt	Host In	0xA990	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	Output DMA	✓	✓	x	1 = trigger on completion of output DMA
1…31	Reserved	✓	✓	x	

Notes:

[7] Logic Op register readback is via the main register only

Config2D

Name	Type	Offset	Format
Config2D	Global *Control register*	0xB618	Bitfield

Bits	Name	Read	Write	Reset	Description
0	Opaque Span	✓	✓	x	In *RasterizerMode*, *AreaStippleMode*, *LogicalOpMode*, *FBWriteMode*, *TextureReadMode*.
1	MultiRXBlit	✗	✗	x	Reserved
2	UserScissorEnable	✓	✓	x	*ScissorMode*
3	FBDestReadEnable	✓	✓	x	In *FBDestReadMode* bit 3 = (ReadEnable)
4	AlphaBlendEnable	✓	✓	x	In *AlphaBlendColorMode* and *AlphaBlendAlphaMode*: bit 4 = AlphaBlendEnable (Enable)
5	DitherEnable	✓	✓	x	In *DitherMode*: bit 5 = DitherEnable (Enable)
6	ForgroundLogicalOpEnable	✓	✓	x	In *LogicalOpMode*: bit 6 = ForgroundLogicalOpEnable (Enable)
7…10	ForgroundLogicalOp	✓	✓	x	In *LogicalOpMode*: Bits 7-10 = ForgroundLogicalOp (LogicOp)
11	BackgroundLogicalOpEnable	✓	✓	x	In *LogicalOpMode*: Bit 11 = BackgroundLogicalOpEnable (Background En.)
12…15	BackgroundLogicalOp	✓	✓	x	In *LogicalOpMode*: Bits 12-15 = BackgroundLogicalOp
16	UseConstantSource	✓	✓	x	In *LogicalOpMode*: bit 16 = UseConstantSource
17	FBWriteEnable	✓	✓	x	In *FBWriteMode*: bit 17 = FBWriteEnable (WriteEnable)
18	Blocking	✓	✓	x	In *FBSourceReadMode* bit 18 = Blocking
19	ExternalSourceData	✓	✓	x	In *FBSourceReadMode* bit 19 = ExternalSourceData
20	LUTMode Enable	✓	✓	x	In *LUTMode*: bit 20 = Enable

Notes: This register updates the mode registers in multiple units as shown. The name in brackets is the field name in the corresponding mode register, if different to the field name for the *Config2D* command. Also note that bit 0 affects several mode registers.

Constant Color

Name	**Type**	**Offset**	**Format**
ConstantColor	Delta *Control register*	0x87E8	Bitfield

Bits	Name	Read	Write	Reset	Description
0…7	Red	✓	✓	x	
8…15	Green	✓	✓	x	
16…23	Blue	✓	✓	x	
24…31	Alpha	✓	✓	x	

Notes: This register holds the constant color in packed format. This is a legacy register maintained for backwards compatibility which has been superceded by the *ConstantColorDDA* register.

The *ConstantColorDDA* register, as well as loading up the constant color register, also loads the DDA start register from the corresponding color byte and sets the dx and dyDom gradients to zero. This allows a constant color to be set up irrespective of the shading mode.

ConstantColorDDA

Name	**Type**	**Offset**	**Format**
ConstantColorDDA	Color *Control register*	0xAFB0	Bitfield

Bits	Name	Read	Write	Reset	Description
0..7	R	✓	✓	x	Red
8..15	G	✓	✓	x	Green
16..23	B	✓	✓	x	Blue
24..31	A	✓	✓	x	Alpha

Notes: This register holds the constant color in packed format. As well as loading up the constant color register it also loads up the DDA start register from the corresponding color byte and sets the dx and dyDom gradients to zero. This allows a constant color to be set up irrespective of the shading mode.

ContextData

Name	Type	Offset	Format
ContextData	Global *Control register*	0x8DD0	Variable

Bits	Name	Read	Write	Reset	Description
1…15	Reserved				
16…31	ContextData	✓	✕	x	Undefined, returned by ContextDump command = (number of data words) -1

Notes: The context data is read from the Host Out FIFO and stored in memory in a context buffer (excluding any tags), while the context mask is typically discarded. This context buffer can be restored by prefixing it with the three words: ***RestoreContext*** tag, context mask (used to generate the buffer in the first place) and the ***ContextData*** tag, and loading it all. The ***ContextData*** tag has the upper 16 bits set to the number of words of context data in the buffer minus one[8]. The layout of the data in the context dump buffer is not important (and is in fact largely undocumented) because no massaging of the data is necessary before it can be restored.

[8] A tag with a count in the upper 16 bits is a hold mode tag so all the subsequent data is automatically given the same tag.

ContextDump

Name	Type	Offset	Format
ContextDump	Global *Command*	0x8DC0	Bitfield

Bits	Name	Read	Write	Reset	Description	Data Words
0	GeneralControl	✕	✓	x	Vertex list and Delta setup mode registers	4
1	Geometry	✕	✓	x	Delta unit state	67
2	Matrices	✕	✓	x	unused	
3	Material	✕	✓	x	unused	
4	Lights0_7	✕	✓	x	unused	
5	Lights8_15	✕	✓	x	unused	
6	RasterPos	✕	✓	x	unused	
7	CurrentState	✕	✓	x	unused	
8	TwoD	✕	✓	x	State used for 2D operations and 2D setup	7
9	DMA	✕	✓	x	State used for tag-driven DMAs (If using Command DMA)	52 (51)
10	Select	✕	✓	x	unused	
11	RasterizerState	✕	✓	x	General setup of the rasterization units	231
12	DDA	✕	✓	x	DDA Values	69
13	Ownership	✕	✓	x	Stripe ownership state	2
14	FogTable	✕	✓	x	Contents of the Fog Table	64
15	LUT	✕	✓	x	Contents of the LUT	256
16	TextureManagement	✕	✓	x	State used for logical texturing (virtual texturing)	9
17…31	Reserved	0	0	x		

Notes: This command forces the P3 to dump the selected context. Context switching can be done on any command boundary but not during internal processing or texture/image downloads. The context is dumped from each unit by the *ContextDump* command and restored by the *ContextRestore* command. The data sent with this command (the context mask) dictates what subset of the full context is to be dumped:

- The context for each unit is defined by the ContextMask sent in the data word of the *ContextDump* and *ContextRestore* commands.
- It appears in the Host Output FIFO tagged as *ContextData* where the host of the output DMA controller can read it.
- The amount of data sent depends on the context mask sent with the command.
- The last tag and data sent to the FIFO is the *ContextDump* tag and mask, but this is not included in the word counts above
- For paired context dump and restore operations the same mask is required.
- The context data is read from the Host Out FIFO and stored in memory in a context buffer (excluding any tags).
- For further information see the *ContextRestore, EndofFeedback, FilterMode and ContextData* registers

ContextRestore

Name	Type	Offset	Format
ContextRestore	Global *Command*	0x8DC8	Bitfield

Bits	Name	Read	Write	Reset	Description	Data Words
0	GeneralControl	✕	✓	x	Vertex list and Delta setup mode registers	4
1	Geometry	✕	✓	x	Delta unit state	67
2	Matrices	✕	✓	x	unused	
3	Material	✕	✓	x	unused	
4	Lights0_7	✕	✓	x	unused	
5	Lights8_15	✕	✓	x	unused	
6	RasterPos	✕	✓	x	unused	
7	CurrentState	✕	✓	x	unused	
8	TwoD	✕	✓	x	State used for 2D operations and 2D setup	7
9	DMA	✕	✓	x	State used for tag-driven DMAs (If using Command DMA)	52 (51)
10	Select	✕	✓	x	unused	
11	RasterizerState	✕	✓	x	General setup of the rasterization units	231
12	DDA	✕	✓	x	DDA Values	69
13	Ownership	✕	✓	x	Stripe ownership state	2
14	FogTable	✕	✓	x	Contents of the Fog Table	64
15	LUT	✕	✓	x	Contents of the LUT	256
16	TextureManagement	✕	✓	x	State used for logical texturing (virtual texturing)	9
17…31	Reserved	0	0	x		

Notes:

- The context for each unit is defined by the ContextMask sent in the data word of the *ContextDump* and *ContextRestore* commands. The various fields in the mask and their effect on units is as shown.
- For further information see the ContextDump, EndofFeedback, FilterMode and ContextData registers

Continue

Name	Type	Offset	Format
Continue	Rasterizer *Command*	0x8058	Integer

Bits	Name	Read	Write	Reset	Description
0…15	Scanlines	✓	✓	x	16 bit unsigned integer
16…31	Reserved	0	0	x	Reserved for future use, mask to 0

Notes: Continues rasterisation to continue after new delta value(s) have been loaded, but doesn't cause either of the trapezoid's edge DDAs to be reloaded. The data field holds the number of scanlines (or sub scanlines) to fill as a 16 bit unsigned integer. Note: this count does not get loaded into the *Count* register.

ContinueNewDom

Name	Type	Offset	Format
ContinueNewDom	Rasterizer *Command*	0x8048	Integer

Bits	Name	Read	Write	Reset	Description
0…15	Scanlines	✓	✓	x	16 bit unsigned integer
16…31	Reserved	0	0	x	Reserved for future use, mask to 0

Notes: This command causes rasterization to continue with a new dominant edge. The dominant edge DDA in the rasterizer is reloaded with the new parameters. The subordinate edge is carried on from the previous trapezoid. This allows any convex 2D polygon to be broken down into a collection of trapezoids and continuity maintained across boundaries.
Since this command only affects the rasterizer DDA (and not any of the other units), it is not suitable for 3D operations.
The data field holds the number of scanlines (or sub scanlines) to fill. Note this count does not get loaded into the ***Count*** register.

ContinueNewLine

Name	Type	Offset	Format
ContinueNewLine	Rasterizer *Command*	0x8040	Integer

Bits	Name	Read	Write	Reset	Description
0…15	Scanlines	✓	✓	x	16 bit unsigned integer
16…31	Reserved	0	0	x	Reserved for future use, mask to 0

Notes: Allows the raterization to continue for the next segment in a polyline. The XY position is carried on from the previous line, however the fraction bits in the DDAs can be kept, set to zero or half under control of the *RasterizerMode*.

The data field holds the number of scanlines (or sub scanlines) to fill as a 16 bit unsigned integer. Note this count does not get loaded into the *Count* register.

The use of *ContinueNewLine* is not recommended for OpenGL because the DDA units will start with a slight error as compared with the value they would have been loaded with for the second and subsequent segments.

ContinueNewSub

Name	Type	Offset	Format
ContinueNewSub	Rasterizer *Command*	0x8050	Integer

Bits	Name	Read	Write	Reset	Description
0…15	Scanlines	✓	✓	x	16 bit unsigned integer
16…31	Reserved	0	0	x	Reserved for future use, mask to 0

Notes: This command causes rasterization to continue with a new subordinate edge. The subordinate edge DDA in the rasterizer is reloaded with the new parameters. The dominant edge is carried on from the previous trapezoid. This is very useful when scan converting triangles with a “knee” (i.e. two subordinate edges. The data field holds the number of scanlines (or sub scanlines) to fill. Note this count does not get loaded into the *Count* register.

Count

Name	**Type**	**Offset**	**Format**
Count	Rasterizer *Control register*	0x8030	Integer

Bits	Name	Read	Write	Reset	Description
0…15	variable	✓	✓	x	16 bit unsigned integer
16…31	Reserved	0	0	x	Reserved for future use, mask to 0

Notes: Mode set in Render command:

- Number of pixels in a line.
- Number of scanlines in a trapezoid.
- Number of sub scanlines in an antialiased trapezoid.
- Diameter of a point in sub scanlines. Unsigned 16 bits.

dAdx

Name	**Type**	**Offset**	**Format**
dAdx	Color *Control register*	0x87D0	Fixed point

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	Unused	0	0	x	

Notes: Used to set the X derivative for the Alpha value for the interior of a trapezoid when in Gouraud shading mode. The format is 24 bit 2's complement 9.15 fixed point numbers. With dBdx, dGdx and dRdx, holds the X gradient values for the Red, Green, Blue and Alpha Color components. See also dFdx for Fog rendering coefficient.

dAdyDom

Name	Type	Offset	Format
dAdyDom	Color DDA *Control register*	0x87D8	Fixed point

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	Unused	0	0	x	

Notes: This register is used to set the Y derivative dominant for the Alpha value along a line, or for the dominant edge of a trapezoid, when in Gouraud shading mode. The value is in 24 bit 2's complement 9.15 fixed point format.

dBdx

Name	Type	Offset	Format
dBdx	Color *Control register*	0x87B8	Fixed point

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	Unused	0	0	x	

Notes: Used to set the X derivative for the Red value for the interior of a trapezoid when in Gouraud shading mode. The format is 24 bit 2's complement 9.15 fixed point numbers.

dBdyDom

Name	Type	Offset	Format
dBdyDom	Color ***Control register***	0x87C0	Fixed point

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	Unused	0	0	x	

Notes: This register is used to set the Y derivative dominant for the Blue value along a line, or for the dominant edge of a trapezoid, when in Gouraud shading mode. The value is in 24 bit 2's complement 9.15 fixed point format.

DeltaControl
DeltaControlAnd
DeltaControlOr

Name	Type	Offset	Format
DeltaControl	Delta	0x9350	Bitfield
DeltaControlAnd	Delta	0xAB20	Bitfield Logic Mask
DeltaControlOr	Delta	0xAB28	Bitfield Logic Mask
	Control Register		

Bits	Name	Read[9]	Write	Reset	Description
0	WrapS	✓	✓	x	1 = enable wrapping in S
1	WrapT	✓	✓	x	1 = enable wrapping in T
2	FullScreenAA	✓	✓	x	1 = enabled
3	DrawLineEndP	✓	✓	x	1 = enabled
4	ForceQ	✓	✓	x	0 = leave Q as delivered, 1 = set Q to 1.0
5	Reserved	0	0	x	
6	UseProvokingV	✓	✓	x	1 = enabled
7	Reserved	0	0	x	
8	WrapS1	✓	✓	x	1 = enable wrapping in S for texture 1
9	WrapT1	✓	✓	x	1 = enable wrapping in T for texture 1
10	ShareQ	✓	✓	x	1 = Set Q1 = Q
11	Line2D	✓	✓	x	1 = draw 2D lines
12	ShareS	✓	✓	x	1 = set S1 = S
13	ShareT	✓	✓	x	1 = set T1 = T
14	ShareColor	✓	✓	x	1 = set diffuse to color
15	Reserved	0	0	x	
16	Reserved	0	0	x	
17-31	Reserved	0	0	x	

Notes:
1. The texture coordinates can be modified by enabling wrapping in S or T. This mode adjusts the texture coordinates so that shortest path is taken; if the normalized S coordinates of two points are 0.1 and 0.9, the shortest path goes from 0.1 to 0, wraps around to 1.0 amd goes down to 0.9.
2. Full screen antialiasing is acheived by drawing at 2x reolution in X and Y, then filtering down to the correct size. This mode requires all X and Y values to be doubled.
3. The end point of a line is not normally drawn, but will be if enabled in this register.
4. If *UseProvokingVertex* is enabled, certain parameters (defined by the *ProvokingVertexMask*) are flat shaded using the vertex specified by the provoking vertex register.

The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

[9] Logic Op register readback is via the main register only

DeltaMode
DeltaModeAnd
DeltaModeOr

Name	**Type**	**Offset**	**Format**
DeltaMode	Delta	0x9300	Bitfield
DeltaModeAnd	Delta	0xAAD0	Bitfield Logic Mask
DeltaModeOr	Delta	0xAAD8	Bitfield Logic Mask
	Control registers		

Bits	Name	Read [10]	Write	Reset	Description
0, 1	TargetChip	✓	✓	x	Read only field, fixed at 1 = TX.
2, 3	DepthFormat	✓	✓	x	This field defines the depth format and hence the final format of the depth parameters to be written into the P3. The options are: 0 = 15 bits 1 = 16 bits 2 = 24 bits 3 = 32 bits
4	FogEnable	✓	✓	x	When set enables the fog calculations. This is qualified by the FogEnable bit in the Draw command.
5	Texture Enable	✓	✓	x	When set enables the texture calculations. This is qualified by the TextureEnable bit in the Draw command.
6	Smooth Shading Enable	✓	✓	x	When set enables the color calculations.
7	Depth Enable	✓	✓	x	When set enables the depth calculations.
8	Specular Texture Enable	✓	✓	x	When set enables the specular texture calculations.
9	Diffuse Texture Enable	✓	✓	x	When set enables the diffuse texture calculations
10	SubPixelCorrectionEnable	✓	✓	x	When set provides the subpixel correction in Y. This is qualified by the SubPixelCorrectionEnable in the Draw command.
11	DiamondExit	✓	✓	x	When set enables the application of the OpenGL 'Diamond-exit' rule to modify the start and end coordinates of lines.
12	NoDraw	✓	✓	x	When set prevents any rendering from starting after the set up calculations are done and parameters sent to P3. This only effect the Draw* commands.
13	ClampEnable	✓	✓	x	When set causes the input values to be clamped into a parameter specific range. Note that the texture parameters are not included.
14, 15	Texture Parameter Mode	✓	✓	x	These field causes the texture parameters to be: 0: Used as given 1: Clamped to lie in the range -1.0 to 1.0 2: Normalise to lie in the range -1.0 to 1.0
16	Reserved	0	0	x	

10 Logic Op register readback is via the main register only

17	BackfaceCull	✓	✓	x	When set enables backface culling. Rejection is based on the sign of the area of the triangle, whether +ve or -ve is controlled by the draw command.
18	ColorOrder	✓	✓	x	Specifies order of colors when packed as RGBA in a 32 bit word, reading from MSB to LSB: 0 = Alpha, Blue, Green, Red 1 = Alpha, Red, Green, Blue Each color component is 8 bits.
19	Bias Coordinates	✓	✓	x	0 = off, 1 = on
20	Reserved	0	0	x	
21-25	Reserved	0	0	x	
26	Texture Enable1	✓	✓	x	0 = off, 1 = on
27	Reserved	✓	✓	x	Reserved
28	Reserved	0	0	x	
29	Texture3D	✓	✓	x	0 = off, 1 = on
30,31	Reserved	0	0	x	

Notes: The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

Depth

Name	**Type**	**Offset**	**Format**
Depth	Depth *Control register*	0x89A8	Integer

Bits	**Name**	**Read**	**Write**	**Reset**	**Description**
0…30	Depth value	✓	✓	x	Integer value right-justified to LSB end and padded with 0s to 31 bits.
31	Reserved	0	0	x	

Notes: Holds an externally sourced 31 bit depth value. If the depth buffer holds less than 31bits then the user supplied depth value is right justified to the least significant end. The unused most significant bits should be set to zero.

This is used in the draw pixels function where the host supplies the depth values through the Depth register. Alternatively this is used when a constant depth value is needed, for example, when clearing the depth buffer, or for 2D rendering where the depth is held constant.

DepthMode
DepthModeAnd
DepthModeOr

Name	Type	Offset	Format
DepthMode	Depth	0x89A0	Bitfield
DepthModeAnd	Depth	0xAC70	Bitfield Logic Mask
DepthModeOr	Depth	0xAC78	Bitfield Logic Mask
	Control registers		

Bits	Name	Read [11]	Write	Reset	Description
0	Enable	✓	✓	x	This bit, when set, enables the depth test and the replacement depth value to depend on the outcome of the test. Otherwise the test always passes and the depth data in the local buffer is not changed.
1	WriteMask	✓	✓	x	This bit, when set enables the depth value in the local buffer to be updated when doing a read-modify-write operation. The byte enables (LB Write) can also be used when the Z value is 16 or 24 bits in size.
2…3	NewDepth Source	✓	✓	x	The depth value to write to the local buffer can come from several places. The options are: 0 = DDA. 1 = Source depth (i.e. read from Local Buffer) 2 = Depth register 3 = LBSourceData register. Only generated when source and destination reads are enabled.
4…6	Compare Function	✓	✓	x	This field selects the compare function to use. The options are: 0 = Never 1 = Less 2 = Equals 3 = Less Equals 4 = Greater 5 = Not Equal 6 = Greater Equal 7 = Always
7…8	Width	✓	✓	x	This field holds the width in bits of the depth field in local buffer. The options are: 0 = 16 bits wide 1 = 24 bits wide 2 = 31 bits wide 3 = 15 bits wide
9	Normalise	✓	✓	x	This bit, when set, will use all 50 bits of the DDA for Z interpolation, even for 24 or less bits of depth. The Width field must be set up to restrict the number of bits used in the comparison operation. When this bit is clear the depth test is compatible with GLINT MX. This bit should be 0 if NonLinearZ is set.
10	NonLinearZ	✓	✓	x	This bit, when set, enables the 32 bit DDA Z value to be encoded in 15, 16 or 24 bits using a non linear pseudo floating point representation. The non linear format is controlled by the following two fields.

[11] Logic Op register readback is via the main register only

11…12	Exponent Scale	✓	✓	x	This field defines how much the exponent should be scaled by. The options are: 0 = scale by 1 1 = scale by 2 2 = scale by 4 3 = scale by 8
13…14	Exponent Width	✓	✓	x	This field defines the number of bits in the depth word to use as exponent bits. The options are: 0 = 1 bit wide exponent field 1 = 2 bits wide 2 = 3 bits wide 3 = 4 bits wide
15…31	Unused	0	0	x	

Notes: The register defines Depth operation. It controls the comparison of a fragment's depth value and updating of the depth buffer. (If the compare function is LESS and result = TRUE then the fragment value is less than the source value.)

The logic operator equivalents behave the same way but the new mode is AND’d or OR’d with the former mode before replacing it.

dFdx

Name	Type	Offset	Format
dFdx	Fog *Control register*	0x86A8	Fixed point

Bits	Name	Read	Write	Reset	Description
0…21	Fraction	✓	✓	x	
22…31	Integer	✓	✓	x	

Notes: Used to set the X derivative for the Fog value for trapezoid rendering. The format is 32 bit 2's complement 10.22 fixed point numbers.

dFdyDom

Name	Type	Offset	Format
dFdyDom	Fog *Control register*	0x86B0	Fixed point

Bits	Name	Read	Write	Reset	Description
0…21	Fraction	✓	✓	x	
22…31	Integer	✓	✓	x	

Notes: This register holds the Y gradient values along the dominant edge for the Fog. The format is 32 bit 2's complement fixed point numbers in 10.22 format

dGdx

Name	Type	Offset	Format
dGdx	Color *Control register*	0x87A0	Fixed point

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	Unused	0	0	x	

Notes: Used to set the X derivative for the Green value for the interior of a trapezoid when in Gouraud shading mode. The format is 24 bit 2's complement 9.15 fixed point numbers.

dGdyDom

Name	Type	Offset	Format
dGdyDom	Color *Control register*	0x87A8	Fixed point

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
23…31	Reserved	0	0	x	Unused

Notes: This register is used to set the Y derivative dominant for the Green value along a line, or for the dominant edge of a trapezoid, when in Gouraud shading mode. The value is in 2's complement 24 bit 9.15 fixed point format.

DitherMode
DitherModeAnd
DitherModeOr

Name	Type	Offset	Format
DitherMode	Global	0x8818	Bitfield
DitherModeAnd	Global	0xACD0	Bitfield Logic Mask
DitherModeOr	Global	0xACD8	Bitfield Logic Mask
	Control Register		

Bits	Name	Read [12]	Write	Reset	Description
0	Enable	✓	✓	x	When set causes the fragment's color values to be dithered or rounded under control of the remaining bits in this register. If this bit is clear then the fragment's color is passed unchanged.
1	Dither Enable	✓	✓	x	When this bit is set any RGB format color is dithered, otherwise it is rounded to the destination size under control of the RoundingMode field. See the table below for the dither matrix and how it is combined with the color components. Color Index formats are always rounded.
2…5	Color Format	✓	✓	x	The color format which in turn is coded from the size and position of the red, green, blue and (if present) the alpha components.
6…7	Xoffset	✓	✓	x	This offset is added to the fragment's x coordinate to derive the x address in the dither table. This allows window-relative dithering using screen coordinates.
8…9	Yoffset	✓	✓	x	This offset is added to the fragment's y coordinate to derive the y address in the dither table. This allows window-relative dithering using screen coordinates.
10	Color Order	✓	✓	x	Holds the color order. The options are: 0 = BGR 1 = RGB
11…13	Reserved	0	0	x	
14	Alpha Dither	✓	✓	x	This bit allows the alpha channel to be rounded even when the color channels are dithered. This helps when antialiasing. 0 = Alpha value is dithered (if DitherEnable is set) 1 = Alpha value is always rounded.
15…16	Rounding Mode	✓	✓	x	0 = Truncate 1 = Round Up 2 = Round Down
17…31	Unused	0	0	x	

[12] Logic Op register readback is via the main register only

Notes: Dithering controls color formatting. The dither function converts the internal color format into the framebuffer color information format.

The following table shows the different color formats supported by the dither unit:

- In the R, G, B and A columns the nomenclature n@m means this component is n bits wide and starts at bit position m in the framebuffer. The least significant bit position is 0 and a dash in a column indicates that this component does not exist for this mode. When two entries are shown the colour value is replicated into both fields.
- Two color ordering formats are supported, namely ABGR and ARGB, with the right most letter representing the color in the least significant part of the word. This is controlled by the Color Order bit in the DitherMode register, and is easily implemented by just swapping the R and B components before conversion into the framebuffer format.
- The only exception to this are the 3:3:2 formats where the actual bit fields sent to the framebuffer data need to be modified as well because the R and B components are differing widths.
- CI processing is not affected by this swap.

				Internal Colour Channels			
	Format	Colour Order	Name	R	G	B	A
	0	BGR	8:8:8:8	8@0	8@8	8@16	8@24
	1	BGR	4:4:4:4	4@0	4@4	4@8	4@12
C	2	BGR	5:5:5:1	5@0	5@5	5@10	1@15
o	3	BGR	5:6:5	5@0	6@5	5@11	-
l	4	BGR	3:3:2	3@0	3@3	2@6	-
o	0	RGB	8:8:8:8	8@16	8@8	8@0	8@24
u	1	RGB	4:4:4:4	4@8	4@4	4@0	4@12
r	2	RGB	5:5:5:1	5@10	5@5	5@0	1@15
	3	RGB	5:6:5	5@11	6@5	5@0	-
	4	RGB	3:3:2	3@5	3@2	2@0	-
CI	15	X	CI8	8@0	8@8	8@16	8@24

The format to use is held in the DitherMode register.

In CI mode the lower byte (CI8) replicated up to the full 32 bit width as an aid to double buffering when the alternative buffers are stored in different bit planes in the same 32 bit word. The replication is done after dithering.

dKdBdx

Name	Type	Offset	Format
dKdBdx	Texture Color *Control register*	0x8D38	Fixed point

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	reserved	0	0	x	

Notes: *dKdBdx* holds the X gradient value for the Blue Kd color component. The format is 24 bit 2's complement fixed point numbers in 9.15 format.

dKdBdyDom

Name	Type	Offset	Format
dKdBdyDom	Texture *Control register*	0x8D40	Fixed point

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	Reserved	0	0	x	

Notes: dKdBdyDom holds the Y gradient value along the dominant edge for the Blue Kd (diffuse) color component. The format is 24 bit 2's complement fixed point numbers in 9.15 format.

dKdGdx

Name	Type	Offset	Format
dKdGdx	Texture Color *Control register*	0x8D20	Fixed point

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	Unused	0	0	x	

Notes: *dKdGdx* holds the X gradient value for the Green Kd (diffuse) color component. The format is 24 bit 2's complement fixed point numbers in 9.15 format.

dKdGdyDom

Name	Type	Offset	Format
dKdGdyDom	Texture *Control register*	0x8D28	Fixed point

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	Unused	0	0	x	

Notes: The Ks and Kd DDA units are responsible for generating the specular and diffuse RGB values. *dKdGdyDom* holds the Y gradient value along the dominant edge for the Green Kd (diffuse) color component. The format is 24 bit 2's complement fixed point numbers in 9.15 format.

dKdRdx

Name	Type	Offset	Format
dKdRdx	Texture *Control register*	0x8D08	Fixed point

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	Unused	0	0	x	

Notes: *dKdRdx* holds the X gradient value for the Red Kd (diffuse) color component. The format is 24 bit 2's complement fixed point numbers in 9.15 format.

dKdRdyDom

Name	Type	Offset	Format
dKdRdyDom	Texture *Control register*	0x8D10	Fixed point

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	Unused	0	0	x	

Notes: *dKdRdyDom* holds the Y gradient value along the dominant edge for the Red Kd (diffuse) color component. The format is 24 bit 2's complement fixed point numbers in 9.15 format.

dKsBdx

Name	Type	Offset	Format
dKsBdx	Texture *Control register*	0x8CB8	Fixed point

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	Unused	0	0	x	

Notes: *dKsBdx* holds the X gradient value for the Blue Ks (specular) color component. The format is 24 bit 2's complement fixed point numbers in 9.15 format. (Note: numeric format differs from the MX.)

dKsBdyDom

Name	Type	Offset	Format
dKsBdyDom	Texture *Control register*	0x8CC0	Fixed point

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	unused	0	0	x	

Notes: *dKsBdyDom* holds the Y gradient value along the dominant edge for the Blue Ks (Specular) color component. The format is 24 bit 2's complement fixed point numbers in 9.15 format.

dKsdx

Name	Type	Offset	Format
dKsdx	Texture *Control register*	0x86D0	Fixed point

Bits	Name	Read	Write	Reset	Description
0…21	Fraction	✓	✓	x	2’s complement 2.22 fixed point fraction
22…23	Integer	✓	✓	x	
24…31	Unused	0	0	x	

Notes: Ks (specular) derivative for unit X. The value is 2.22 2's complement format..

dKsdyDom

Name	Type	Offset	Format
dKsdyDom	Texture *Control register*	0x86D8	Fixed point

Bits	Name	Read	Write	Reset	Description
0…21	Fraction	✓	✓	x	
22…23	Integer	✓	✓	x	
24…31	Unused	0	0	x	

Notes: Ks (specular) derivative per unit Y along the dominant edge. The value is 2.22 2's complement format

dKsGdx

Name	Type	Offset	Format
dKsGdx	Texture *Control register*	0x8CA0	Fixed point

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	Unused	0	0	x	

Notes: *dKsGdx* holds the X gradient value for the Green Ks (specular) color component. The format is 24 bit 2's complement fixed point numbers in 9.15 format. (Note: numeric format differs from MX.)

dKsGdyDom

Name	**Type**	**Offset**	**Format**
dKsGdyDom	Texture *Control register*	0x8CA8	Fixed point

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	Unused	0	0	x	

Notes: *dKsGdyDom* holds the Y gradient value along the dominant edge for the Green Ks (Specular) color component. The format is 24 bit 2's complement fixed point numbers in 9.15 format.

dKsRdx

Name	**Type**	**Offset**	**Format**
dKsRdx	Texture *Control register*	0x8C88	Fixed point

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	Unused	0	0	x	

Notes: dKsRdx holds the X gradient value for the Re Ks (specular) color component. The format is 24 bit 2's complement fixed point numbers in 9.15 format. (Note: numeric format has changed from the MX.)

dKsRdyDom

Name	Type	Offset	Format
dKsRdyDom	Texture *Control register*	0x8CC0	Fixed point

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	Unused	0	0	x	

Notes: dKsRdyDom holds the Y gradient value along the dominant edge for the Red Ks (Specular) color component. The format is 24 bit 2's complement fixed point numbers in 9.15 format.

DMAAddr

Name	Type	Offset	Format
DMAAddr	Input Control Register	0xA980	Integer

Bits	Name	Read	Write	Reset	Description
0…1	Reserved	0	0	X	
2…31	Address	✓	✓	X	Address

Notes: This register holds the byte address of the next DMA buffer to read from (reading doesn't start until the *DMACount* command). The bottom two bits of the address are ignored, hence the byte address is forced to be 32 bit aligned.

This register should not be confused with the PCI register of the same name. *DMAAddr* must be loaded by itself and not as part of any increment, hold or indexed group. See also: *DMACount*.

DMAContinue

Name	Type	Offset	Format
DMAContinue	Input *Command*	0xA9F8	Integer

Bits	Name	Read	Write	Reset	Description
0…29	Count	✓	✓	x	Number of DMA words to transfer
30…31	Reserved	0	0	x	

Notes:

DMACount

Name	Type	Offset	Format
DMACount	Input *Control register*	0xA988	Integer

Bits	Name	Read	Write	Reset	Description
0…29	Count	✓	✓	x	Number of DMA words to transfer
30…31	Reserved	0	0	x	

Notes: At chip reset the MasterEnable bit in the *CFGCommand* register must be set to allow DMA to operate. Then, for the simplest form of DMA, the host software prepares a host buffer containing register address tag descriptions and data values. The host writes the base address of this buffer to the *DMAAddr* register and the count of the number of words to transfer to the *DMACount* register. Writing to the *DMACount* register starts the DMA transfer and the host is then free to perform other work.

DMAFeedback

Name	Type	Offset	Format
DMAFeedback	Input *Command*	0xAA10	Integer

Bits	Name	Read	Write	Reset	Description
0…29	Count	✓	✓	x	Number of DMA words to transfer
30…31	Reserved	0	0	x	Reserved

Notes: The Feedback DMA mechanism allows the collection and transfer of an unspecified amount of data from the Host Out FIFO. This can be used for OpenGL feedback and select modes.

- The feedback DMA transfer is set up by using the *DMAOutputAddress* register and the *DMAFeedback* command.
- The *DMAOutputAddress* holds the address where the data is to be written The start address is given as a byte address but the lower two bits are ignored.
- The *DMAFeedback* command with the length of the memory buffer (in words) is sent to start the Output DMA controller. Data is never written beyond the end of the given buffer length.
- Once all the data to write to memory has been generated the *EndOfFeedback* command is sent to terminate the DMA operation. A count of the number of words transferred is recorded in the PCIFeedbackCount register.

DMAMemoryControl

Name	**Type**	**Offset**	**Format**
DMAMemoryControl	Input *Command*	0xB780	Bitfield

Bits	Name	Read	Write	Reset	Description
0	InputDMA Memory	✓	✓	x	0 = PCI, 1 = AGP
1	Reserved	0	0		
2	Input DMA Alignment	✓	✓	x	0 = off, 1 = on
3	Index Memory	✓	✓	x	0 = PCI, 1 = AGP
4	Reserved	0	0	x	
5	Index Alignment	✓	✓	x	0 = off, 1 = on
6	Vertex Memory	✓	✓	x	0 = PCI, 1 = AGP
7	Reserved	0	0	x	
8	Vertex Alignment	✓	✓	x	0 = off, 1 = on
9	ReadDMA Memory	✓	✓	x	0 = PCI, 1 = AGP
10	Reserved	0	0	x	
11	ReadDMA Alignment	✓	✓	x	0 = off, 1 = on
12-23	Reserved	0	0	x	
24-28	Burst Size	✓	✓	x	
29-30	Reserved	0	0	x	
31	WriteDMA Alignment	✓	✓	x	0 = off, 1 = on

Notes:

DMAOutputAddress

Name	Type	Offset	Format
DMAOutputAddress	Input *Command*	0xA9E0	Integer

Bits	Name	Read	Write	Reset	Description
0…1	Reserved	0	0	x	Reserved
2…31	Address	✓	✓	x	32 bit aligned address

Notes: This register holds the byte address where the output DMA controller will write to. The lower two bits of the address are ignored. This register must be loaded by itself and not as part of any increment, hold or indexed group.

DMAOutputCount

Name	Type	Offset	Format
DMAOutputCount	Input *Command*	0xA9E8	Integer

Bits	Name	Read	Write	Reset	Description
0..29	Count	✓	✓	x	Number of DMA words to transfer
30…31	Reserved	0	0	x	

Notes: This command starts a new output DMA if the output DMA controller is idle, otherwise it will block until the output DMA controller becomes available and all subsequent commands and register loads are suspended.

- The number of words to read from the P3 Host Out FIFO is given in the bottom 24 bits of the command, and the memory buffer address will have previously been set up in the *DMAOutputAddress* register.
- The P3 FilterMode register must have been set up to allow the required tags and/or data to be written in to the FIFO..
- This register must be loaded by itself and not as part of any increment, hold or indexed group.
- See also: *DMAOutputAddress*

DMARectangleRead

Name	**Type**	**Offset**	**Format**
DMARectangleRead	Input *Control Register*	0xA9A8	Bitfield

Bits	Name	Read	Write	Reset	Description
0-11	Width	✓	✓	x	Width of the rectangle in pixels. Range 0…4095
12-23	Height	✓	✓	x	Height of the rectangle in pixels. Range 0…4095
24-25	PixelSize	✓	✓	x	The size of the pixels in the source image to read. The pixel size is used during alignment and packing. The values are: 0 = 8 bits, 1 = 16 bits, 2 = 24 bits, 3 = 32 bits
26	Pack	✓	✓	x	This field, when set, causes the data to be packed into 32 bit words when used, otherwise the data is right justified and any unused bits (in the most significant end of the word) are set to zero.
27-28	ByteSwap	✓	✓	x	These bits control the byte swapping of the data read from the PCI bus before it is aligned and packed/unpacked. If the input bytes are labeled ABCD on input then they are swapped as follows: 0 = ABCD (i.e. no swap) 1 = BADC 2 = CDAB 3 = DCBA
29	Reserved	0	0	x	
30-31	Alignment	✓	✓	x	When set, causes P3 to start and stop PCI or AGP transfers on 64 byte boundaries where possible.

Notes:
1. The Rectangle DMA mechanism allows image data to be transferred from host memory to the P3. The image data may be a sub image of a larger image and have any natural alignment or pixel size. Information regarding the rectangle transfer is held in registers loaded from the input FIFO or a DMA buffer.
2. The pixel data read from host memory is always packed, however when passed to P3 it can be in packed or unpacked format. It can also, optionally, be aligned on 64 byte boundaries.
3. The minimum number of PCI reads are used to align and pack the image data.
4. P3 is set up to rasterize the destination area for the pixel data (depth, stencil, color, etc.) with *SyncOnHostData* or *SyncOnBitMask* enabled in the Render command. This is done before the Rectangular DMA is started.
5. This register must be loaded by itself and not as part of any increment, hold or indexed group.
6. See also *DMARectangleReadAddress; DMARectangleReadLinePitch; DMARectangleReadTarget.*

DMARectangleReadAddress

Name	Type	Offset	Format
DMARectangleReadAddress	Input *Control Register*	0xA9B0	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Address	✓	✓	x	32 bit pixel aligned address

Notes: This register provides the byte address of the first pixel in the image or sub image to read during a rectangular DMA transfer from host memory to P3. The address should be aligned to the natural size of the pixel, except for 24 bit pixels which may be aligned to any byte boundary. This register must be loaded by itself and not as part of any increment, hold or indexed group.
See also: *DMARectangleRead; DMARectangleReadLinePitch; DMARectangleReadTarget*

DMARectangleReadLinePitch

Name	Type	Offset	Format
DMARectangleReadLine Pitch	Input *Control Register*	0xA9B8	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Line Pitch	✓	✓	x	LinePitch

Notes: This register defines the amount, in bytes, to move from one scanline in the image to the next scanline during a rectangular DMA transfer from host memory to P3. For a sub image this is based on the width of the whole image. The pitch is held as a 32 bit 2's complement number. This is normally an integer multiple of the number of bytes in a pixel. The register must be loaded by itself and not as part of any increment, hold or indexed group.
See also: *DMARectangleReadAddress; DMARectangleRead; DMARectangleReadTarget.*

DMARectangleReadTarget

Name	**Type**	**Offset**	**Format**
DMARectangleReadTarget	Input *Command*	0xA9C0	Bitfield

Bits	Name	Read	Write	Reset	Description
0-10 Tag	Tag	✓	✓	x	Tag to use with DMA data.
11-31	Reserved	0	0	x	Reserved

Notes:

1. This register holds the 16 bit tag sent to the Rasterizer just before the image data is sent during a rectangular DMA transfer from host memory to the P3. Normally it would be one of the tags allowed by the rasterizer during a SyncOnHostData or SyncOnBitMask operation with the tag mode set to Hold. The secondary PCI bus traffic is minimized by sending multiple image words with a single tag (with a count).
2. This register must be loaded by itself and not as part of any increment, hold or indexed group.
3. See also: *DMARectangleReadAddress; DMARectangleReadLinePitch; DMARectangleRead*

DMARectangleWrite

Name	Type	Offset	Format
DMARectangleWrite	Input *Control register*	0xA9C8	Bitfield

Bits	Name	Read	Write	Reset	Description
0-11	Width	✓	✓	x	Width of the rectangle in pixels. Range 0…4095
12-23	Height	✓	✓	x	Height of the rectangle in pixels. Range 0…4095
24-25	PixelSize	✓	✓	x	The size of the pixels in the source image to read. The pixel size is used during alignment and packing. The values are: 0 = 8 bits, 1 = 16 bits, 2 = 24 bits, 3 = 32 bits
26	Pack	✓	✓	x	This field, when set, specifies the data is right justified and any unused bits (in the most significant end of the word) are set to zero. Otherwise the data read from the Host Out FIFO is packed. N.B. this is the inverse of the bit setting in GLINT Gamma.
27-28	ByteSwap	✓	✓	x	These bits control the byte swapping of the data written to the PCI bus. If the input bytes are labeled ABCD on input then they are swapped as follows: 0 = ABCD (i.e. no swap) 1 = BADC 2 = CDAB 3 = DCBA
29	Reserved	0	0	x	
30-31	Alignment	✓	✓	x	When set, causes P3 to start and stop PCI or AGP transfers on 64 byte boundaries where possible.

Notes:
1. The Rectangle DMA mechanism allows image data to be transferred from P3 to host memory. The image data may be a sub image of a larger image and have any natural alignment or pixel size. Information regarding the rectangle transfer is held in registers loaded from the input FIFO or a DMA buffer. Note that failure to supply an EOF may have unpredictable results.
2. The pixel data written to host memory is always packed, however when read from the Host Out FIFO it can be in packed or unpacked format. Note that it is packed when Reset. It can also, optionally, be aligned on 64 byte boundaries.
3. The minimum number of PCI writes are used to align and pack the image data.
4. P3 is set up to rasterize the source area for the pixel data (depth, stencil, color, etc.) enabled in the Render command. This is done before the Rectangular DMA is started.
5. This register must be loaded by itself and not as part of any increment, hold or indexed group.
6. See also: *DMARectangleReadAddress; DMARectangleReadLinePitch; DMARectangleReadTarget*

DMARectangleWriteAddress

Name	Type	Offset	Format
DMARectangleWrite Address	Input	0xA9D0	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…31	Address	✓	✓	x	32 bit pixel aligned address

Notes:

- This register provides the byte address of the first pixel in the image or sub image to write during a rectangular DMA transfer from P3 to host memory. The address should be aligned to the natural size of the pixel, except for 24 bit pixels which may be aligned to any byte boundary.
- This register must be loaded by itself and not as part of any increment, hold or indexed group.
- See also: *DMARectangleWrite; DMARectangleWriteLinePitch; DMAReadGLINTSource*

DMARectangleWriteLinePitch

Name	Type	Offset	Format
DMARectangleWriteLine Pitch	Input	0xA9D8	Integer
	Control Register		

Bits	Name	Read	Write	Reset	Description
0…31	Line Pitch	✓	✓	x	LinePitch

Notes: This register defines the amount, in bytes, to move from one scanline in the image to the next scanline during a rectangular DMA transfer from P3 to host memory. For a sub image this is based on width of the whole image.

- The pitch is held as a 32 bit 2's complement number. This is normally an integer multiple of the number of bytes in a group.
- See also: *DMARectangleWriteAddress; DMARectangleWrite; DMAReadGLINTSource*

DownloadAddress

Name	Type	Offset	Format
DownloadAddress	Framebuffer ***Control register***	0xB0D0	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Page Address	✓	✓	x	32 bit integer value from 0 to 65535

Notes: This register holds the address to which to download 32 bits of data. The address is incremented after every write. The simplest way to download data to the framebuffer (or indeed any memory) is to use the **DownloadAddress** message to set up the word address. Each subsequent **DownloadData** command sends 32 bits of message data to the download address, after which the download address is auto incremented to address the next word. The bottom two bits of the **DownloadAddress** are forced to zero for the memory update, and readback will return the incremented address value

DownloadData

Name	Type	Offset	Format
DownloadData	Framebuffer ***Control register***	0xB0D8	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Data	✕	✓	x	32 bit data

Notes: This register holds the data to write to memory. The address will have previously been set up using the DownloadAddress message. Each **DownloadData** command sends 32 bits of message data to the download address, after which the download address is auto incremented to address the next word. The bottom two bits of the **DownloadAddress** are forced to zero for the memory update, and readback returns the incremented address value

DownloadGlyphWidth

Name	Type	Offset	Format
DownloadGlyphWidth	Setup *Control register*	0xB658	Integer

Bits	Name	Read	Write	Reset	Description
0…15	Glyph width	✓	✓	x	16 bit integer value from 0 to 65535

Notes: This register holds the width of the glyph in bytes (range 0…31) which is just about to be downloaded via the *GlyphData* register. This must be sent for every download as it sets up some state used to manage the download.

DownloadTarget

Name	Type	Offset	Format
DownloadTarget	2DSetup *Control register*	0xB650	Tag name

Bits	Name	Read	Write	Reset	Description
0…12	Tag name	✓	✓	x	

Notes: This tag holds the register the various download operations will write the expanded or generated data to. It can hold any legal tag, but typically will be set to *FBData* or *FBSourceData*.

dQ1dx

Name	Type	Offset	Format
dQ1dx	Texture *Control register*	0x8438	Fixed point

Bits	Name	Read	Write	Reset	Description
0…n	Fraction	✓	✓	x	
n…31	Integer	✓	✓	x	

Notes: dQ1dx holds the X gradient values for the Q1 texture coordinate. The format is 32 bit 2's complement fixed point numbers. The binary point is arbitrary but must be consistent for all S1, T1 and Q1 values.

dQ1dyDom

Name	Type	Offset	Format
dQ1dyDom	Texture *Control register*	0x8440	Fixed point

Bits	Name	Read	Write	Reset	Description
0…n	Fraction	✓	✓	x	
n…31	Integer	✓	✓	x	

Notes: dQ1dyDom holds the Y gradient values along the dominant edge for the Q1 texture coordinate. The format is 32 bit 2's complement fixed point. The binary point is at an arbitrary location, but must be consistent for all S1, T1 and Q1 values.

dQdx

Name	Type	Offset	Format
dQdx	Texture *Control register*	0x83C0	Fixed point

Bits	Name	Read	Write	Reset	Description
0…n	Fraction	✓	✓	x	
n…31	Integer	✓	✓	x	

Notes: Sets the X derivative for the Q parameter for texture map interpolation. The value is in 32 bit 2's complement fixed point format. The binary point is at an arbitrary location, but must be consistent for all S, T and Q values.

dQdy

Name	Type	Offset	Format
dQdy	Texture *Control register*	0x83E8	Fixed point

Bits	Name	Read	Write	Reset	Description
0…n	Fraction	✓	✓	x	
n…31	Integer	✓	✓	x	

Notes: The register holds the Y gradient value for the Q texture coordinate. The format is 32 bit 2's complement fixed point numbers. The binary point is at an arbitrary location, but must be consistent for all S, T and Q values.

dQdyDom

Name	Type	Offset	Format
dQdyDom	Texture *Control register*	0x83C8	Fixed point

Bits	Name	Read	Write	Reset	Description
0…n	Fraction	✓	✓	x	
n…31	Integer	✓	✓	x	

Notes: Sets the Y derivative dominant for the Q parameter for texture map interpolation. Expressed in 32 bit 2's complement fixed point, binary point arbitrary but must be consistent for all S, T and Q values.

DrawLine0

Name	Type	Offset	Format
DrawLine0	Delta *Command*	0x9318	Bitfield

Bits	Name	Read	Write	Reset	Description
0..15	X	✗	✓	x	2’s complement
16..31	Y	✗	✓	x	2’s complement

Notes:

- Initiates a line (between V0 and V1) set up and render. *DrawLine2D01* and *DrawLine2D10* commands have identical behaviour to *Drawline0* and *DrawLine1* and are only duplicated for efficient grouping in DMA.
- *LineCoord0* loads vertex store 0, *LineCoord1* loads vertex store 1. *DrawLine0* draws a line from vertex 0 to vertex1, *DrawLine1* draws a line from vertex 1 to vertex 0.

DrawLine1

Name	Type	Offset	Format
DrawLine01	Delta *Command*	0x9320	Bitfield

Bits	Name	Read	Write	Reset	Description
0..15	X	✗	✓	x	2's complement
16..31	Y	✗	✓	x	2's complement

Notes:
- Initiates a line (between V1 and V0) set up and render. DrawLine2D01 and DrawLine2D10 commands have identical behaviour to Drawline01 and DrawLine10 and are only duplicated for efficient grouping in DMA.
- LineCoord0 loads vertex store 0, LineCoord1 loads vertex store 1. DrawLine01 draws a line from vertex 0 to vertex1, DrawLine10 draws a line from vertex 1 to vertex 0.

DrawLine2D01

Name	Type	Offset	Format
DrawLine2D01	Delta *Command*	0x9778	Bitfield

Bits	Name	Read	Write	Reset	Description
0..15	X	✗	✓	x	2's complement
16..31	Y	✗	✓	x	2's complement

Notes:
- Initiates a line (between V0 and V1) set up and render. *DrawLine2D01* and *DrawLine2D10* commands have identical behaviour to *Drawline1* and *DrawLine1* and are only duplicated for efficient grouping in DMA.
- *LineCoord0* loads vertex store 0, *LineCoord1* loads vertex store 1. *DrawLine0* draws a line from vertex 0 to vertex1, *DrawLine1* draws a line from vertex 1 to vertex 0.

DrawLine2D10

Name	Type	Offset	Format
DrawLine2D01	Delta *Command*	0x9768	Bitfield

Bits	Name	Read	Write	Reset	Description
0..15	X	✗	✓	x	2's complement
16..31	Y	✗	✓	x	2's complement

Notes:
- Initiates a line (between V1 and V0) set up and render. *DrawLine2D01* and *DrawLine2D10* commands have identical behaviour to *Drawline0* and *DrawLine1* and are only duplicated for efficient grouping in DMA.
- LineCoord0 loads vertex store 0, *LineCoord1* loads vertex store 1. *DrawLine0* draws a line from vertex 0 to vertex1, *DrawLine1* draws a line from vertex 1 to vertex 0.

DrawPoint

Name	Type	Offset	Format
DrawPoint	Delta *Command*	0x9330	Bitfield

Bits	Name	Read	Write	Reset	Description
0	AreaStipple Enable	✗	✓	x	Area stippling enable
1	LineStipple Enable	✗	✓	x	Line stippling enable.
2	ResetLine Stipple	✗	✓	x	Reset line stipple counters
3	FastFillEnable	✗	✓	x	Enable span fills
4, 5	Unused	0	0	x	
6, 7	Primitive Type	✗	✓		Select primitive type: 0 = Line 1 = Trapezoid 2 = Point
8	Antialiase Enable	✗	✓		Enables antialiasing
9	Antialiasing Quality	✗	✓		Set (=1) sub pixel resolution to 8x8 Reset (=0) sub pixel resolution to 4x4.
10	UsePoint Table	✗	✓		When this bit and the AntialiasingEnable are set, the dx values used to move from one scanline to the next are derived from the Point Table.
11	SyncOnBit Mask	✗	✓		See *Render command* for details
12	SyncOnHost Data	✗	✓		When this bit is set a fragment is produced only when one of the following registers have been received from the host: *Depth*, *Stencil*, *Color* or *FBData*, *FBSourceData*
13	TextureEnable	✗	✓	x	Enables texturing of the fragments produced during rasterisation.

14	FogEnable	✕	✓	x	Enables fogging of the fragments produced during rasterisation. Note that the Fog Unit must be suitably enabled as well for any fogging to occur.
15	Coverage Enable	✕	✓	x	Enables the coverage value produced as part of the antialiasing to weight the alpha value in the alpha test unit.
16	SubPixel Correction Enable	✕	✓	x	Enables the sub pixel correction of the color, depth, fog and texture values at the start of a scanline.
17	Reserved	0	0	x	Reserved
18	SpanOperation	✕	✓	x	Indicates the writes are to use the constant color found in the previous *FBBlockColor* register.
19…26	Reserved	✕	✕	x	Reserved
27	FBSourceRead Enable	✕	✓	x	Enables source buffer reads to be done in the Framebuffer Read Unit.

Notes: Initiates point set up and render. *Command* - data field duplicates the Render command – for details see the *Render* command description.

DrawTriangle

Name	Type	Offset	Format
DrawTriangle	Delta *Command*	0x9308	Bitfield

Bits	Name	Read	Write	Reset	Description
0	AreaStipple Enable	✕	✓	x	Area stippling enable
1	LineStipple Enable	✕	✓	x	Line stippling enable.
2	ResetLine Stipple	✕	✓	x	Reset line stipple counters
3	FastFillEnable	✕	✓	x	Enable span fills
4, 5	Unused	0	0	x	
6, 7	Primitive Type	✕	✓		Select primitive type: 0 = Line 1 = Trapezoid 2 = Point
8	Antialiase Enable	✕	✓		Enables antialiasing
9	Antialiasing Quality	✕	✓		Set (=1) sub pixel resolution to 8x8 Reset (=0) sub pixel resolution to 4x4.
10	UsePoint Table	✕	✓		When this bit and the AntialiasingEnable are set, the dx values used to move from one scanline to the next are derived from the Point Table.
11	SyncOnBit Mask	✕	✓		See *Render command* for details
12	SyncOnHost Data	✕	✓		When this bit is set a fragment is produced only when one of the following registers have been received from the host: *Depth*, *Stencil*, *Color* or *FBData*, *FBSourceData*

13	TextureEnable	✗	✓	x	Enables texturing of the fragments produced during rasterisation.
14	FogEnable	✗	✓	x	Enables fogging of the fragments produced during rasterisation. Note that the Fog Unit must be suitably enabled as well for any fogging to occur.
15	Coverage Enable	✗	✓	x	Enables the coverage value produced as part of the antialiasing to weight the alpha value in the alpha test unit.
16	SubPixel Correction Enable	✗	✓	x	Enables the sub pixel correction of the color, depth, fog and texture values at the start of a scanline.
17	Reserved	0	0	x	Reserved
18	SpanOperation	✗	✓	x	Indicates the writes are to use the constant color found in the previous *FBBlockColor* register.
19…26	Reserved	✗	✗	x	Reserved
27	FBSourceRead Enable	✗	✓	x	Enables source buffer reads to be done in the Framebuffer Read Unit.

Notes: Initiates a triangle set up and render. P3 Delta unit only.. *Command* - data field duplicates the Render command – for details see the *Render* command description.

dRdx

Name	Type	Offset	Format
dRdx	Color DDA *Control register*	0x8788	Fixed point

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	Unused	0	0	x	

Notes: Used to set the X derivative for the Red value for the interior of a trapezoid when in Gouraud shading mode. The format is 24 bit 2's complement 9.15 fixed point numbers.

dRdyDom

Name	**Type**	**Offset**	**Format**
dRdyDom	Color *Control register*	0x8790	Fixed point

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	Unused	0	0	x	

Notes: This register is used to set the Y derivative dominant for the Red value along a line, or for the dominant edge of a trapezoid, when in Gouraud shading mode. The value is in 2's complement 9.15 fixed point format.

dS1dx

Name	**Type**	**Offset**	**Format**
dS1dx	Texture *Control register*	0x8408	Fixed point

Bits	Name	Read	Write	Reset	Description
0…n	Fraction	✓	✓	x	
n…31	Integer	✓	✓	x	

Notes: dS1dx holds the X gradient value for the S1 texture coordinate. The format is 32 bit 2's complement fixed point numbers. The binary point is at an arbitrary location, but must be consistent for all S1, T1 and Q1 values.

dS1dyDom

Name	Type	Offset	Format
dS1dyDom	Texture *Control register*	0x8410	Fixed point

Bits	Name	Read	Write	Reset	Description
0…n	Fraction	✓	✓	x	
n…31	Integer	✓	✓	x	

Notes: The dominant edge gradient of the texture S1 parameter. The format is 32 bit 2's complement fixed point numbers. The value is in 2's complement fixed point format. The binary point is at an arbitrary location, but must be consistent for all S1, T1 and Q1 values.

dSdx

Name	Type	Offset	Format
DSdx	Texture *Control register*	0x8390	Fixed point

Bits	Name	Read	Write	Reset	Description
0…n	Fraction	✓	✓	x	
n…31	Integer	✓	✓	x	

Notes: Sets the X derivative for the S parameter for texture map interpolation. The value is in 2's complement fixed point format. The binary point is at an arbitrary location, but must be consistent for all S, T and Q values.

dSdy

Name	Type	Offset	Format
DSdy	Texture *Control register*	0x83D8	Fixed point

Bits	Name	Read	Write	Reset	Description
0…n	Fraction	✓	✓	x	
n…31	Integer	✓	✓	x	

Notes: The register holds the Y gradient value for the S texture coordinate. The format is 32 bit 2's complement fixed point numbers. The binary point is at an arbitrary location, but must be consistent for all S, T and Q values.

dSdyDom

Name	Type	Offset	Format
DSdyDom	Texture *Control register*	0x8398	Fixed point

Bits	Name	Read	Write	Reset	Description
0…n	Fraction	✓	✓	x	
n…31	Integer	✓	✓	x	

Notes: Sets the Y derivative dominant for the S parameter for texture map interpolation. Expressed in 2's complement fixed point, binary point arbitrary but must be consistent for all S, T and Q values.

dT1dx

Name	Type	Offset	Format
DT1dx	Texture *Control register*	0x8420	Fixed point

Bits	Name	Read	Write	Reset	Description
0…n	Fraction	✓	✓	x	
n…31	Integer	✓	✓	x	

Notes: dT1dx holds the X gradient value for the T1 texture coordinate. The format is 32 bit 2's complement fixed point numbers. The binary point is at an arbitrary location but must be consistent for all S1, T1 and Q1 values.

dT1dyDom

Name	Type	Offset	Format
DT1dyDom	Texture *Control register*	0x8428	Fixed point

Bits	Name	Read	Write	Reset	Description
0…n	Fraction	✓	✓	x	
n…31	Integer	✓	✓	x	

Notes: The dominant edge gradient of the texture T1 parameter. The format is 32 bit 2's complement fixed point numbers. The value is in 2's complement fixed point format. The binary point is at an arbitrary location, but must be consistent for all S1, T1 and Q1 values.

dTdx

Name	**Type**	**Offset**	**Format**
dTdx	Texture *Control register*	0x83A8	Fixed point

Bits	Name	Read	Write	Reset	Description
0…n	Fraction	✓	✓	x	
n…31	Integer	✓	✓	x	

Notes: Sets the X derivative for the T parameter for texture map interpolation. The value is in 32 bit 2's complement fixed point format. The binary point is at an arbitrary location, but must be consistent for all S, T and Q values.

dTdy

Name	**Type**	**Offset**	**Format**
dTdy	Texture *Control register*	0x83E0	Fixed point

Bits	Name	Read	Write	Reset	Description
0…n	Fraction	✓	✓	x	
n…31	Integer	✓	✓	x	

Notes: The register holds the Y gradient value for the T texture coordinate. The format is 32 bit 2's complement fixed point numbers. The binary point is at an arbitrary location, but must be consistent for all S, T and Q values.

dTdyDom

Name	**Type**	**Offset**	**Format**
dTdyDom	Texture *Control register*	0x83B0	Fixed point

Bits	Name	Read	Write	Reset	Description
0…n	Fraction	✓	✓	x	
n…31	Integer	✓	✓	x	

Notes: Sets the Y derivative dominant for the T parameter for texture map interpolation. Expressed in 2's complement fixed point, binary point arbitrary but must be consistent for all S, T and Q values.

dXDom

Name	Type	Offset	Format
Delta X Dominant	Rasterizer *Control register*	0x8008	Fixed point

Bits	Name	Read	Write	Reset	Description
0…15	Fraction	✓	✗	x	
16…31	Integer	✓	✗	x	

Notes: The gradient for the dominant edge held as a 16.16 fixed point 2s complement value. Value added when moving from one scanline (or sub scanline) to the next for the dominant edge in trapezoid filling. The register also holds the change in X when plotting lines. For Y major lines this will be some fraction (dx/dy), otherwise it is normally ± 1.0, depending on the required scanning direction.

dXSub

Name	Type	Offset	Format
Delta X Subordinate	Rasterizer *Control register*	0x8018	Fixed point

Bits	Name	Read	Write	Reset	Description
0…15	Fraction	✓	✗	x	
16…31	Integer	✓	✗	x	

Notes: The gradient for the subordinate edge: the value added when moving from one scanline or sub scanline to the next for the subordinate edge in trapezoid filling. Two's complement fixed point 16.16 format.

dY

Name	Type	Offset	Format
Delta Y	Rasterizer *Control register*	0x8028	Fixed point

Bits	Name	Read	Write	Reset	Description
0…15	Fraction	✓	✗	x	
16…31	Integer	✓	✗	x	

Notes: The change in Y between scanlines or sub-scanlines: the value added to Y to move from one scanline to the next. For X major lines this will be some fraction (dy/dx), otherwise it is normally ± 1.0, depending on the required scanning direction. Two's complement fixed point 16.16 format.

dZdxL

Name	Type	Offset	Format
dZdxL	Fog *Control register*	0x89C8	Fixed point pair

Bits	Name	Read	Write	Reset	Description
0…15	Reserved	0	0	x	LSBs all 0
16…31	Integer	✓	✓	x	16bit LSB part of 32.16 fixed point value

Notes: *dZdxL* and *dZdxU* set the depth derivative per unit in X used in rendering trapezoids and/or for Fog when Fog mode is UseZ. *dZdxU* holds the 32 most significant bits, and *dZdxL* the least significant 16 bits. The value is in 2's complement 32.16 fixed point format.

dZdxU

Name	Type	Offset	Format
dZdxU	Fog *Control register*	0x89C0	Fixed point pair

Bits	Name	Read	Write	Reset	Description
32…63	dZdxU	✓	✓	x	32 bit integer

Notes: *dZdxL* and *dZdxU* set the depth derivative per unit in X used in rendering trapezoids and/or for Fog when Fog mode is UseZ. *dZdxU* holds the 32 most significant bits, and *dZdxL* the least significant 16 bits. The value is in 2's complement 32.16 fixed point format.

dZdyDomL

Name	Type	Offset	Format
dZdyDomL	Fog *Control register*	0x89D8	Fixed point pair

Bits	Name	Read	Write	Reset	Description
0…15	Reserved	✗	✗	x	LSBs all 0
16…31	Integer	✓	✓	x	16bit LSB part or 32.16 value

Notes: *dZdyDomL* and *dZdyDomU* set the depth derivative per unit in Y along the dominant edge or along a line during trapezoid rendering when Fog mode is "UseZ". *dZdyDomU* holds the most significant bits, and the least significant bits.. The value is in 2's complement 32.16 fixed point format.

dZdyDomU

Name	Type	Offset	Format
dZdyDomU	Fog *Control register*	0x89D0	Fixed point pair

Bits	Name	Read	Write	Reset	Description
32..63	integer	✓	✓	x	32 bit integer part

Notes: *dZdyDomU* and *dZdyDomL* set the depth derivative per unit in Y for the dominant edge, or along a line. *dZdyDomU* holds the most significant bits, and *dZdyDomL* the least significant bits. The value is in 2's complement 32.16 fixed point format.

EndOfFeedback

Name	Type	Offset	Format
EndOfFeedback	Output *Command*	0x8FF8	unused

Bits	Name	Read	Write	Reset	Description
0	EndofFeedback	✗	✓	x	Command tag

Notes: DMA transfers to or from the P3 Host Out FIFO can use either a fixed count (where the precise amount of data is known) or a variable count (where the amount of data is unknown or undefined). EndofFeedback is used to terminate DMA variable-length mode transfers.

Variable Count:
Typically, variable count mode is used for Context Dump or Run Length Encoded data. In this mode the Output DMA controller is placed in Feedback mode and continues to transfer data from the Host Out FIFO until it finds an EndOfFeedback tag.

The FilterMode register should be set up by setting bits 18 and 19 to allow both context data and tags through so tags and data inappropriate to this mode can be discarded and the EndOfFeedback tag can be identified. Bit 20 of the FilterMode register enables RLE data into the output FIFO. The Host Out FIFO does not need to be empty but this would be preferable.

The PCI FeedbackSelectCount register will hold the number of words written to memory when the Output DMA has finished. This method relieves the programmer from knowing beforehand how much context data will be saved.

FBBlockColor

Name	Type	Offset	Format
FBBlockColor	Framebuffer *Control register*	0x8AC8	integer

Bits	Name	Read	Write	Reset	Description
0…31	Color	✓	✓	x	32 bit raw framebuffer format

Notes: Holds the color and optionally alpha value to write during span writes. The data is in raw framebuffer format and is automatically replicated up to 128 bits and loaded into FBBlockColor[0…3]. The local registers as well as the registers in the memory devices are updated. This color information is used for constant color transparent span fills or constant color opaque span fill for foreground pixels. Readback returns the data in *FBBlockcolor0*.

FBBlockColor [0]
FBBlockColor [1]
FBBlockColor [2]
FBBlockColor [3]

Name	Type	Offset	Format
FBBlockColor [0…3]	Framebuffer *Control registers*	0xB060, 0xB068, 0xB070, 0xB078	

Bits	Name	Read	Write	Reset	Description
0…31	Color word 1	✓	✓	x	32 bit raw framebuffer value

Notes: These registers update the corresponding 32 bits of block color (in raw framebuffer format) in the local register and memory devices. This color information is used for constant color transparent span fills or constant color opaque span fill for foreground pixels. Use of the individual registers allows different colors for pattern fills, for example.

FBBlockColorBack

Name	Type	Offset	Format
FBBlockColorBack	Framebuffer *Control register*	0xB0A0	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Color word	✓	✓	x	32 bit raw framebuffer format

Notes: Holds the color and optionally alpha value to write during span writes. The data is in raw framebuffer format and is automatically replicated up to 128 bits. The local registers, FBBlockColorBack[0…3] are updated. This color information is used for constant color transparent span fills or constant color opaque span fill for background pixels. Readback returns the data in *FBBlockcolorBack0*.

FBBlockColorBack [0]
FBBlockColorBack [1]
FBBlockColorBack [2]
FBBlockColorBack [3]

Name	Type	Offset	Format
FBBlockColorBack [0…3]	Framebuffer *Control registers*	0xB080, 0xB088, 0xB090, 0xB098	integer

Bits	Name	Read	Write	Reset	Description
0…31	Color word 1	✓	✓	x	32 bit raw framebuffer value

Notes: These registers update the corresponding 32 bits of block color (in raw framebuffer format) in the local register. This color information is used for constant color transparent span fills or constant color opaque span fill for background pixels.

FBColor

Name	Type	Offset	Format
FBColor	Framebuffer *Control register*	0x8A98	

Bits	Name	Read	Write	Reset	Description
0…31	Reserved	0	✗	x	Reserved

Notes: Internal register used in image upload and processed as configured in FilterMode settings. This register should not be written to. It is documented solely to provied the tag name of the data returned through the Host Out FIFO. Format depends on the raw framebuffer organization and any reformatting which takes place in Color processing

FBDestReadBufferAddr[0…3]

Name	Type	Offset	Format
FBDestReadBufferAddr [0…3]	Framebuffer *Control registers*	0xAE80, 0xAE88, 0xAE90, 0xAE98	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Address	✓	✓	x	32 bit value

Notes: Holds the 32 bit base address of the four destination buffers in memory. The address is a byte address and should be aligned to the natural boundary for the selected pixel size.

FBDestReadBufferOffset[0…3]

Name	Type	Offset	Format
FBDestReadBufferOffset [0…3]	Framebuffer *Control register*	0xAEA0, 0xAEA8, 0xAEB0, 0xAEB8	Integer

Bits	Name	Read	Write	Reset	Description
0…15	X offset	✓	✓	x	2's complement X offset
16…31	Y offset	✓	✓	x	2's complement Y offset

Notes: These registers hold the offset added to the fragment's coordinate for each destination buffer. The new coordinate is used for address calculations. This offset allows, for example, window relative coordinates to be converted into screen relative ones prior to patching (patching only works screen relative).

FBData

Name	Type	Offset	Format
FBSourceData	Framebuffer *Control register*	0x8AA0	Integer

Bits	Name	Read	Write	Reset	Description
0…63	Mask	✓	✗	x	This message holds 64 bits of destination span data.

Notes:

FBDestReadBufferWidth[0…3]

Name	Type	Offset	Format
FBDestReadBufferWidth [0…3]	Framebuffer *Control register*	0xAEC0, 0xAEC8, 0xAED0, 0xAED8	Integer

Bits	Name	Read	Write	Reset	Description
0…11	Width	✓	✓	x	12 bit width of buffer

Notes: Holds the width of each destination buffer. The width is held as a 12 bit unsigned integer so has the range 0…4095.

FBDestReadEnables
FBDestReadEnablesAnd
FBDestReadEnablesOr

Name	**Type**	**Offset**	**Format**
FBDestReadEnables	Framebuffer	0xAEE8	Bitfield
FBDestReadEnablesAnd	Framebuffer	0xAD20	Bitfield Logic Mask
FBDestReadEnablesOr	Framebuffer	0xAD28	Bitfield Logic Mask
	Control registers		

Bits	Name	Read [13]	Write	Reset	Description
0…3	E1 to E3	✓	✓	x	These bits are the Enable bits. Software assigns these to major modes which can be enabled or disabled (such as Alpha Blending) it wants the FB Read Unit to track so destination reads are automatically done when necessary. When a bit is 1 it is enabled. E0…E3 are used for fragments.
4…7	E4 to E7	✓	✓	x	Used for spans
8…11	R0 to R3	✓	✓		These are Read bits. Software assigns these to operations within a major mode which require reads. For example the major mode would be Alpha Blending, but not all alpha blending option require the destination buffer to be read. When a bit is 1 a read is required. R0…R3 are used for fragments.
12…15	R4 to R7	✓	✓	x	Used for spans
24…31	Reference Alpha	✓	✓	x	This is the alpha value used to disable reads when AlphaFiltering is enabled.

Notes: Monitors potential FB Read activity on up to 4 parameters assignable in software. E.g.:
E0 = Alpha Blend Enable
R0 = Set whenever an alpha blend mode requires a read
E1 = logically Enable
R1 = Set whenever a logical operation requires a read

The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

[13] Logic Op register readback is via the main register only

FBDestReadMode
FBDestReadModeAnd
FBDestReadModeOr

Name	**Type**	**Offset**	**Format**
FBDestReadMode	Alpha Blend	0xAEE0	Bitfield
FBDestReadModeAnd	Alpha Blend	0xAC90	Bitfield Logic Mask
FBDestReadModeOr	Alpha Blend	0xAC98	Bitfield Logic Mask
	Control registers		

Bits	Name	Read [14]	Write	Reset	Description
0	ReadEnable	✓	✓	x	This bit, when set, causes fragments or spans to read from the those buffers which are enabled (Enable[0…3] fields). If this bit is clear then no reads from any of the destination buffers are made.
1	Reserved	✗	✗	x	
2…4	Stripe Pitch	✓	✓	x	This field specifies the number of scanlines between the first scanline in a stripe and the first scanline in the next stripe. It would normally be set to number of RXs * StripeHeight. The options are: 0 = 1 4 = 16 1 = 2 5 = 32 2 = 4 6 = 64 3 = 8 7 = 128 This field will normally be set to zero for P3.
5…7	StripeHeight	✓	✓	x	This field specifies the number of scanlines in a stripe. The options are: 0 = 1 3 = 8 1 = 2 4 = 16 2 = 4 This field will normally be set to zero for P3.
8	Enable0	✓	✓	x	Enable reading from buffers 0. The ReadEnable bit must also be set.
9	Enable1	✓	✓	x	Enable reading from buffers 1.
10	Enable2	✓	✓	x	Enable reading from buffers 2.
11	Enable3	✓	✓	x	Enable reading from buffers 3.
12…13	Layout0	✓	✓	x	Selects the layout of the pixel data in memory for buffer 0. The options are: 0 = Linear 1 = Patch64 Color buffer 2 = Patch32_2 Large texture maps 3 = Patch2 Small texture maps Note: 32_2 and Patch2 are not supported for span reads.

[14] Logic Op register readback is via the main register only

14…15	Layout1	✓	✓	x	Selects the layout of the pixel data in memory for buffer 1.
16…17	Layout2	✓	✓	x	Selects the layout of the pixel data in memory for buffer 2.
18…19	Layout3	✓	✓	x	Selects the layout of the pixel data in memory for buffer 3.
20 21 22 23	Origin0 Origin1 Origin2 Origin3	✓	✓	x	These fields selects where the window origin is for buffer 0…3 respectively. The options are: 0 = Top Left. 1 = Bottom Left
24	Blocking	✓	✓	x	This bit, when set, causes destination span reads to block to prevent reads and writes from overlapping (in time). Each span is read in full and then written. This is less efficient than streaming (bit is clear), but allows overlapping blits (spans overlap) without corruption. Note this does not need to be set if the destination read and write buffers are the same.
25	Reserved	0	0	x	
26	UseRead Enables	✓	✓	x	When this bits is set the enables in the FBDestReadEnables register are used to determine if a destination read is required. The ReadEnable bit must also be set and the corresponding buffer bits as well for a read to occur.
27	Alpha Filtering	✓	✓	x	This bit, when set, compares the fragment's alpha value and if it is equal to the AlphaReference value (held in the FBReadEnables register) then no read is done. This is done to save memory bandwidth when the alpha blend mode is such that with the given alpha value the destination color doesn't contribute to the fragment's color.

Notes: The destination address calculation(s) are controlled by the FBDestReadMode register and the address is a function of X, Y, FBDestReadBufferAddr, FBDestReadBufferOffset, FBDestReadBufferWidth and PixelSize parameters. The Addr, Offset and Width are specified independently for each of the four possible write buffers.

The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

FBHardwareWriteMask

Name	Type	Offset	Format
FBHardwareWriteMask	Framebuffer *Control register*r	0x8AC0	

Bits	Name	Read	Write	Reset	Description
0…31	Write mask	✓	✓	x	32 bit mask

Notes: This register holds the write mask used for all writes. When a bit is set the corresponding bit in each framebuffer word is set (enabled for writing). The masking is actually done in the memory devices so has zero impact on performance and doesn't require any reads.

- The hardware write mask applies only where the memory devices (i.e. SGRAM) are used. Where it is not supported, this register should not be written to.
- Where hardware writemask is supported and used, the software writemask must be disabled by setting all bits to 1.
- If the framebuffer is used in 8bit packed mode the hardware writemask must be 8 bits wide and replicated to all four bytes of this register.

FBSoftwareWriteMask

Name	Type	Offset	Format
FBSoftwareWriteMask	Framebuffer *Control register*r	0x8820	int

Bits	Name	Read	Write	Reset	Description
0…31	Write mask	✓	✓	x	32 bit mask

Notes: Contains the software writemask for the framebuffer:

- If a bit is set (=1) then the corresponding bit in the framebuffer is enabled for writing.
- If hardware writemasking is implemented then the software writemask must be disabled by setting all bits to 1.
- Framebuffer destination reads should be enabled if the write mask is ***not*** set to all ones.

FBSourceData

Name	Type	Offset	Format
FBSourceData	Framebuffer *Control register*	0x8AA8	Integer

Bits	Name	Read	Write	Reset	Description
0…63	Mask	✓	✗	x	This message hold the 32 bits of source pixel data when generated by an active step. When generated for span masking it holds 64 bits of source span data.

Notes:

FBSourceReadBufferAddr

Name	Type	Offset	Format
FBSourceReadBufferAddr	Framebuffer *Control register*	0xAF08	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Address	✓	✓	x	32 bit value

Notes: This register holds the 32 bit base address of the source buffer in memory. The address is a byte address and should be aligned to the natural boundary for the selected pixel size.

FBSourceReadBufferOffset

Name	Type	Offset	Format
FBDestReadBufferOffset	Framebuffer *Control register*	0xAF10	Integer

Bits	Name	Read	Write	Reset	Description
0…15	X offset	✓	✓	x	2's complement X offset
16…31	Y offset	✓	✓	x	2's complement Y offset

Notes: These registers hold the offsets added to the fragment's coordinate for each destination buffer. The new coordinate is used for address calculations. This offset allows, for example, window relative coordinates to be converted into screen relative ones prior to patching (patching only works screen relative).

FBSourceReadBufferWidth

Name	Type	Offset	Format

FBSourceReadBufferWidth Framebuffer *Control register* 0xAF18 Integer

Bits	Name	Read	Write	Reset	Description
0…11	Width	✓	✓	x	12 bit buffer width

Notes: This register holds the width of the source buffer. The width is held as a 12 bit unsigned integer so has the range 0…4095.

FBSourceReadMode
FBSourceReadModeAnd
FBSourceReadModeOr

Name	Type	Offset	Format
FBSourceReadMode	Framebuffer	0xAF00	Bitfield
FBSourceReadModeAnd	Framebuffer	0xACA0	Bitfield
FBSourceReadModeOr	Framebuffer	0xACA8	Bitfield
	Control register		

Bits	Name	Read [15]	Write	Reset	Description
0	ReadEnable	✓	✓	x	This bit, when set, causes fragments or spans to read from the source buffer providing they are enabled in the *Render command* (using the FBSourceReadEnable bit, bit 27). If this bit is clear then no source reads are made.
1	Reserved	×	×	x	
2…4	StripePitch	✓	✓	x	This field specifies the number of scanlines between the first scanline in a stripe and the first scanline in the next stripe. It would normally be set to number of RXs * StripeHeight. The options are: 0 = 1 4 = 16 1 = 2 5 = 32 2 = 4 6 = 64 3 = 8 7 = 128 This field will normally be set to zero for P3.
5…7	Stripe Height	✓	✓	x	This field specifies the number of scanlines in a stripe. The options are: 0 = 1 3 = 8 1 = 2 4 = 16 2 = 4 This field will normally be set to zero for P3.
8…9	Layout	✓	✓	x	This field selects the layout of the pixel data in memory for buffer 0…3 respectively. The options are: 0 = Linear 1 = Patch64 Color buffer 2 = Patch32_2 Large texture maps 3 = Patch2 Small texture maps Note Patch32_2 and Patch2 are not supported for span reads.
10	Origin	✓	✓	x	This field selects where the window origin is. The options are: 0 = Top Left. 1 = Bottom Left

[15] Logic Op register readback is via the main register only

11	Blocking	✓	✓	x	This bit, when set, causes source span reads to block to prevent reads and writes from overlapping (in time). Each span is read in full and then written. This is less efficient than streaming (bit is clear), but allows overlapping blits (spans overlap) without corruption.
12	Reserved	×	×	x	
13	UseTexel Coord	✓	✓	x	This bit, when set, allows the texel coordinate generated in the Texture Read Unit to be used instead of the fragments X, Y coordinate as part of the source address calculation. The Texture Read Unit must also be set up as appropriate, although failure to do so will not cause a chip hang. This bit should not be set when span reads are done. This is useful for stretch blits when the source is the framebuffer.
14	WrapX Enable	✓	✓	x	This bit, when set, causes the X coordinate to be wrapped. The wrapping is done on power of two pixel boundaries as defined in the WrapX field. When span reads are used the wrapping point must be a multiple of 16 bytes so smaller patterns must be replicated in X to be this width. Normal pixel reads do not suffer from this restriction.
15	WrapY Enable	✓	✓	x	This bit, when set, causes the Y coordinate to be wrapped. The wrapping is done on power of two pixel boundaries as defined in the WrapY field.
16…19	WrapX	✓	✓	x	This field defines the mask to use for X wrapping. The options are: 0…9 mask = $2^{(WrapX + 1)} - 1$ 10…15 mask = 0xffff
20…23	WrapY				This field defines the mask to use for Y wrapping. The options are: 0…9 mask = $2^{(WrapY + 1)} - 1$ 10…15 mask = 0xffff
24	External Source Data				This bit, when set, indicates that even though source reads are disabled source data is being provided from an external source. This will be data downloaded by the host (using the Color command) or from the LUT. This data is interleaved with the destination data as if the source data had really been read from memory. This is important for span logical op processing when the source data is *not* from memory.
25…31	Unused	0	0	x	

Notes: Distinct source reads are still needed when a source image is to be blended or logically combined into the destination buffer or buffers.

The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

FBWriteBufferAddr[0…3]

Name	Type	Offset	Format
FBWriteBufferAddr[0…3]	Framebuffer *Control registers*	0xB000, 0xB008, 0xB010, 0xB018	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Address	✓	✓	x	32 bit value

Notes: These registers holds the 32 bit base addresses of the four buffers in memory. The address is a byte address and should be aligned to the natural boundary for the selected pixel size

FBWriteBufferOffset[0…3]

Name	Type	Offset	Format
FBWriteBufferOffset[0…3]	Framebuffer *Control registers*	0xB020, 0xB028, 0xB030, 0xB038	Integer

Bits	Name	Read	Write	Reset	Description
0…15	X offset	✓	✓	x	2's complement X offset
16…31	Y offset	✓	✓	x	2's complement Y offset

Notes: These registers hold the offset added to the fragment's coordinate for each buffer. The new coordinate is used for address calculations. This offset allows, for example, window relative coordinates to be converted into screen relative ones prior to patching (patching only works screen relative).

FBWriteBufferWidth[0…3]

Name	Type	Offset	Format
FBWriteBufferWidth[0…3]	Framebuffer	0xB040, 0xB048, 0xB050, 0xB058	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…11	Width	✓	✓	x	12 bit width of buffer

Notes: These registers hold the width of each buffer. The width is held as a 12 bit unsigned integer so has the range 0…4095

FBWriteMode
FBWriteModeAnd
FBWriteModeOr

Name	Type	Offset	Format
FBWriteMode	Alpha Blend	0x8AB8	Bitfield
FBWriteMode And	Alpha Blend	0xACF0	Bitfield Logic Mask
FBWriteMode Or	Alpha Blend	0xACF8	Bitfield Logic Mask
	Control registers		

Bits	Name	Read [16]	Write	Reset	Description
0	WriteEnable	✓	✓	x	This bit, when set, causes fragment or spans to write to the buffer 0, or if mulit-reads in FB Read then write are done to the corresponding buffers which were read. If this bit is clear then no writes to any buffer are made. Note that the Enable[0…3] bits are ignored unless Replicate is also set.
1…3	reserved	✓	✓	x	
4	Replicate	✓	✓	x	This bit, when set, causes each fragment or span to be written into all the enabled buffers. It should not be set if multi-buffer reads are enabled in FB Read Mode.

[16] Logic Op register readback is via the main register only

5	OpaqueSpan	✓	✓	x	This field determines how constant color spans are written (recall the Render command selects between constant color or variable color spans). The options are: 0 = Transparent 1 = Opaque Transparent spans just use one color for the foreground pixels and the background pixels are not written. Opaque spans write to foreground and background pixels using *FBBlockColor* for the foreground pixels and *FBBlockColorBack* for the background pixels.
6…8	StripePitch	✓	✓	x	This field specifies the number of scanlines between the first scanline in a stripe and the first scanline in the next stripe. It would normally be set to number of RXs * StripeHeight. The options are: 0 = 1 4 = 16 1 = 2 5 = 32 2 = 4 6 = 64 3 = 8 7 = 128 This field will normally be set to 0 for P3.
9…11	StripeHeight	✓	✓	x	This field specifies the number of scanlines in a stripe. The options are: 0 = 1 3 = 8 1 = 2 4 = 16 2 = 4 This field will normally be set to 0 for P3.
12 13 14 15	Enable0 Enable1 Enable2 Enable3	✓	✓	x	These bits, when set, enable writes to buffer 0…3 respectively during replication. The WriteEnable bit must also be set.
16…17 18…19 20…21 22…23	Layout0 Layout1 Layout2 Layout3	✓	✓	x	These fields select the layout of the pixel data in memory for buffer 0…3 respectively. The options are: 0 = Linear 1 = Patch64 Color buffer 2 = Patch32_2 Large texture maps 3 = Patch2 Small texture maps
24 25 26 27	Origin0 Origin1 Origin2 Origin3	✓	✓	x	These fields select where the window origin is for buffer 0…3 respectively. The options are: 0 = Top Left. 1 = Bottom Left
28..31	Unused	0	0	x	

Notes: The Framebuffer is responsible for:

- Managing the updates to up to 4 memory buffers,
- Calculating the write address(es) of the fragment in the memory,
- Combining multiple fragments in the same memory word,
- Calculating the write addresses of the spans in the memory,
- Aligning span data and issuing multiple normal writes,
- Implementing transparent or opaque fills,
- Dispatch the addresses and data/mask to the Memory Controller .

The FBWriteMode command controls write operations.

The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

FeedbackX

Name	Type	Offset	Format
FeedbackX	Output *Control register*	0x8F88	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Runlength	✗	✓	x	32 bit integer value

Notes: This tag is used to hold the run length when run length encoding of image data is enabled.

FeedbackY

Name	Type	Offset	Format
FeedbackY	Output *Control register*	0x8F90	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Runlength	✓	✓	x	32 bit integer value

Notes: This tag is used to hold the run length when run length encoding of image data is enabled.

FillBackgroundColor

Name	Type	Offset	Format
FillBackgroundColor	2DSetup *Control register*	0x8330	Integer

Bits	Name	Read	Write	Reset	Description
0...31	Background Color	✗	✓	x	32 bit integer

Notes: *FillBackgroundColor* is an alias for the *BackGroundColor* register. With *ForegroundColor*, holds the foreground and background color values. A background pixel is a pixel whose corresponding bit in the color mask is zero. The color format is in the raw framebuffer format and 8 or 16 bit pixels are automatically replicated to fill the 32 bits of register.

FillConfig2D0
FillConfig2D1

Name	**Type**	**Offset**	**Format**
FillConfig2D0	2DSetup	0x8338	Bitfield
FillConfig2D1	2DSetup	0x8360	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	Opaque Span	✗	✓	x	In *RasterizerMode*, *AreaStippleMode*, *LogicalOpMode*, *FBWriteMode*, *TextureReadMode*.
1	MultiRXBlit	✗	✓	x	*RasterizerMode*, *ScissorMode*
2	UserScissorEnable	✗	✓	x	*ScissorMode*
3	FBDestReadEnable	✗	✓	x	In *FBDestReadMode* bit 3 = (ReadEnable)
4	AlphaBlendEnable	✗	✓	x	In *AlphaBlendColorMode* and *AlphaBlendAlphaMode*: bit 4 = AlphaBlendEnable (Enable)
5	DitherEnable	✗	✓	x	In *DitherMode*: bit 5 = DitherEnable (Enable)
6	ForgroundLogicalOpEnable	✗	✓	x	In *LogicalOpMode*: bit 6 = ForgroundLogicalOpEnable (Enable)
7…10	ForgroundLogicalOp	✗	✓	x	In *LogicalOpMode*: Bits 7-10 = ForgroundLogicalOp (LogicOp)
11	BackgroundLogicalOpEnable	✗	✓	x	In *LogicalOpMode*: Bit 11 = BackgroundLogicalOpEnable (Background En.)
12…15	BackgroundLogicalOp	✗	✓	x	In *LogicalOpMode*: Bits 12-15 = BackgroundLogicalOp
16	UseConstantSource	✗	✓	x	In *LogicalOpMode*: bit 16 = UseConstantSource
17	FBWriteEnable	✗	✓	x	In *FBWriteMode*: bit 17 = FBWriteEnable (WriteEnable)
18	Blocking	✗	✓	x	In *FBSourceReadMode* bit 18 = Blocking
19	ExternalSourceData	✗	✓	x	In *FBSourceReadMode* bit 19 = ExternalSourceData
20	LUTMode Enable	✗	✓	x	In *LUTMode*: bit 20 = Enable
21…31	Unused	0	0	x	

Notes: *FillConfig2D0* and *FillConfig2D1* are aliases for the *Config2D* register. This register updates the mode registers in multiple units as shown. The name in brackets is the field name in the corresponding mode register, if different to the field name for the *Config2D* command. Also note that bit 0 affects several mode registers.

FillFBDestReadBufferAddr0

Name	Type	Offset	Format
FillFBDestReadBufferAddr0	Framebuffer *Control register*	0x8310	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Address	✗	✓	x	32 bit value

Notes: An alias for FBDestReadBufferAddr0, this register holds the 32 bit base address of the destination buffer in memory. The address is a byte address and should be aligned to the natural boundary for the selected pixel size.

FillFBSourceReadBufferAddr

Name	Type	Offset	Format
FillFBSourceReadBuffer Addr	2DSetup *Control register*	0x8308	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Address	✗	✓	x	32 bit value

Notes: This register is an alias for *FBSourceReadBufferAddr* and holds the 32 bit base address of the source buffer in memory. The address is a byte address and should be aligned to the natural boundary for the selected pixel size.

FillFBSourceReadBufferOffset0

Name	Type	Offset	Format
FillFBDestReadBuffer Offset0	2DSetup *Control register*	0x8340	Integer

Bits	Name	Read	Write	Reset	Description
0…15	X offset	✓	✓	x	2’s complement X offset
16…31	Y offset	✓	✓	x	2’s complement Y offset

Notes: Aliasing the *FillFBDestReadBufferOffset*0 register, this register holds the offset added to the fragment's coordinate for each destination buffer. The new coordinate is used for address calculations. This offset allows, for example, window relative coordinates to be converted into screen relative ones prior to patching (patching only works screen relative).

FillFBWriteBufferAddr0

Name	**Type**	**Offset**	**Format**
FillFBWriteBuffer Addr0	2DSetup *Control register*	0x8300	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Address	✗	✓	x	32 bit value

Notes: Aliasing for the *FBWriteBufferAddr0* registers, this register holds the 32 bit base addresses of the buffer in memory. The address is a byte address and should be aligned to the natural boundary for the selected pixel size

FillForegroundColor0

Name	**Type**	**Offset**	**Format**
FillForegroundColor0	2DSetup *Control register*	0x8328	Integer

Bits	Name	Read	Write	Reset	Description
0...31	Foreground Color	✗	✓	x	32 bit integer

Notes: This registers is an alias for the *ForegroundColor* register. With *BackgroundColor*, holds the foreground and background color values. The color format is in the raw framebuffer format and 8 or 16 bit pixels are automatically replicated to fill the 32 bits of register.

FillForegroundColor1

Name	**Type**	**Offset**	**Format**
FillForegroundColor1	2DSetup *Control register*	0x8358	Integer

Bits	Name	Read	Write	Reset	Description
0...31	Foreground Color	✗	✓	x	32 bit integer

Notes: This register is an alias for the *ForegroundColor* register. With *BackgroundColor*, holds the foreground and background color values. The color format is in the raw framebuffer format and 8 or 16 bit pixels are automatically replicated to fill the 32 bits of register.

FillGlyphPosition

Name	**Type**	**Offset**	**Format**
FillGlyphPosition	2DSetup *Control register*	0x8368	Integer

Bits	Name	Read	Write	Reset	Description
0…15	X offset	✕	✓	x	2's complement X coordinate
16…31	Y offset	✕	✓	x	2's complement Y coordinate

Notes: This register is an alias for the *GlyphPosition* register. It defines the glyph origin for use by the *Render2Dglyph* command.

FillRectanglePosition

Name	**Type**	**Offset**	**Format**
FillRectanglePosition	2DSetup *Control register*	0x8348	Integer

Bits	Name	Read	Write	Reset	Description
0…15	X offset	✕	✓	x	2's complement X coordinate
16…31	Y offset	✕	✓	x	2's complement Y coordinate

Notes: This is an alias for the *RectanglePosition* register. It defines the rectangle origin for use by the Render2D command.

FillRender2D

Name	**Type**	**Offset**	**Format**
FillRender2D	2DSetup *Control register*	0x8350	Bitfield

Bits	Name	Read	Write	Reset	Description
0…11	Width	✗	✓	x	Specifies the width of the rectangle in pixels. Its range is 0…4095.
12…13	Operation	✗	✓	x	This two bits field is encoded as follows: 0 = Normal 1 = SyncOnHostData 2 = SyncOnBitMask 3 = PatchOrderRendering The SyncOnHostData and SyncOnBitMask settings just set the corresponding bit in the Render command. PatchOrderRendering decomposes the input rectangle in to a number of smaller rectangels to make better use of the page structure of patched memory (see later).
14	FBReadSource	✗	✓	x	This bit sets the FBReadSourceEnable bit in the Render command.
15	SpanOperation	✗	✓	x	This bit sets the SpanOperation bit in the Render command.
16…27	Height	✗	✓	x	Specifies the height of the rectangle in pixels. Its range is 0…4095.
28	IncreasingX	✗	✓	x	This bit, when set, specifies the rasterisation is to be done in increasing X direction.
29	IncreasingY	✗	✓	x	This bit, when set, specifies the rasterisation is to be done in increasing Y direction.
30	AreaStipple	✗	✓	x	This bit sets the AreaStippleEnable bit in the Render command.
31	Texture	✗	✓	x	This bit sets the TextureEnable bit in the Render command.

Notes: This command starts a rectangle being rendered from the origin given by the RectanglePosition register.

FillScissorMaxXY

Name	Type	Offset	Format
FillScissorMaxXY	2DSetup *Control register*	0x8320	Fixed point

Bits	Name	Read	Write	Reset	Description
0…15	X coordinate	✗	✓	x	2's complement fixed point X coordinate
16…31	Y coordinate	✗	✓	x	2's complement fixed point Y coordinate

Notes: This register is an alias for ScissorMaxXY. It holds the maximum XY scissor coordinate - i.e. the rectangle corner farthest from the screen origin.

FillScissorMinXY

Name	Type	Offset	Format
FillScissorMinXY	2DSetup *Control register*	0x8318	Fixed point

Bits	**Name**	**Read**	**Write**	**Reset**	**Description**
0…15	X coordinate	✗	✓	x	2's complement fixed point X coordinate
16…31	Y coordinate	✗	✓	x	2's complement fixed point Y coordinate

Notes: This register is an alias for the *ScissorMinXY* register. It holds the minimum XY scissor coordinate - i.e. the rectangle corner closest to the screen origin.

FilterMode
FilterModeAnd
FilterModeOr

Name	Type	Offset	Format
FilterMode	Output	0x8C00	Bitfield
FilterModeAnd	Output	0xAD00	Bitfield Logic Mask
FilterModeOr	Output	0xAD08	Bitfield Logic Mask
	Control registers		

Bits	Name	Read [17]	Write	Reset	Description
0…3	Reserved	✓	✓	x	Reserved for diagnostic use – set to 0
4	LBDepthTag	✓	✓	x	When set allows the *LBDepth* tag to be written into the output FIFO.
5	LBDepthData	✓	✓	x	When set allows the data upload from the Depth buffer to be written into the output FIFO.
6	StencilTag	✓	✓	x	When set allows the LBStencil tag to be written into the output FIFO.
7	StencilData	✓	✓	x	When set allows the data upload from the Stencil buffer to be written into the output FIFO.
8	FBColorTag	✓	✓	x	When set allows the *FBColor* tag to be written into the output FIFO.
9	FBColorData	✓	✓	x	When set allows the data upload from the framebuffer to be written into the output FIFO.
10	SyncTag	✓	✓	x	When set allows Sync tag to be written into the output FIFO.
11	SyncData	✓	✓	x	When set allows the Sync data to be written into the output FIFO.
12	StatisticsTag	✓	✓	x	When set allows the *PickResult, MaxHitRegion* and *MinHitRegion* tags to be written into the output FIFO.
13	StatisticsData	✓	✓	x	When set allows the *PickResult, MaxHitRegion* and *MinHitRegion* data to be written into the output FIFO.
14	RemainderTag	✓	✓	x	When set allows any tags not covered by the categories in this table to be written into the output FIFO.
15	RemainderData	✓	✓	x	When set allows any data not covered by the categories in this table to be written into the output FIFO.
16…17	ByteSwap	✓	✓		This field controls the byte swapping of the data field when it is written into the output FIFO. The options are: 0 = ABCD (i.e. no swap) 1 = BADC 2 = CDAB 3 = DCBA

[17] Logic Op register readback is via the main register only

18	ContextTag	✓	✓	x	When set allows the *ContextData* and *EndOfFeedback* tags to be written into the output FIFO.
19	ContextData	✓	✓	x	When set allows the ContextData and *EndOfFeedback* data to be written into the output FIFO.
20	RunLength Encode Data	✓	✓	x	This bit, when set, will write run length encoded data into the host out FIFO.
21…31	Unused	0	0	x	

Notes: This register can only be updated if the *Security* register is set to 0.

The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

FlushSpan

Name	**Type**	**Offset**	**Format**
FlushSpan	Rasterizer *Command*	0x8060	Tag

Bits	Name	Read	Write	Reset	Description
0…31	Reserved	✗	0	x	Reserved for future use

Notes: Causes any partial sub scanlines to be written out - command used when antialiasing to force rasterization of any remaining subscanlines in a primitive.

FlushWriteCombining

Name	**Type**	**Offset**	**Format**
FlushWriteCombining	Input *Control register*	0x8910	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Reserved	✗	✓	x	32 bit value

Notes:

FogColor

Name	**Type**	**Offset**	**Format**
FogColor	Fog *Control register*	0x8698	Bitfield

Bits	Name	Read	Write	Reset	Description
0…7	Red	✓	✓	x	Red
8…15	Green	✓	✓	x	Green
16…23	Blue	✓	✓	x	Blue
24…31	Reserved	0	0	x	Reserved

Notes: This register holds the fog color to interpolate with.

FogMode
FogModeAnd
FogModeOr

Name	Type	Offset	Format
FogMode	Fog	0x8690	Bitfield
FogModeAnd	Fog	0xAC10	Bitfield Logic Mask
FogModeOr	Fog	0xAC18	Bitfield Logic Mask
	Control registers		

Bits	Name	Read [18]	Write	Reset	Description
0	Enable	✓	✓	x	This bit, when set, and qualified by the FogEnable bit in the *Render* command causes the current fragment color to be modified by the fog coefficient and background color.
1	ColorMode	✓	✓	x	This bit selects the color mode. The two options are: 0 = RGB. The RGB fog equation is used. 1 = CI. The Color Index fog equation is used.
2	Table	✓	✓	x	This bit, when set, causes the Fog Index to be mapped via the FogTable before it controls the blending between the fragment's color and the fog color, otherwise the DDA value is used directly.
3	UseZ	✓	✓	x	This bit, when set, causes the DDA to be loaded with the Z DDA values instead of the Fog DDA values. It also adjusts the clamping of the DDA output.
4…8	ZShift	✓	✓	x	This field specifies the amount the (z from DDA + zBias) is right shifted by before it is clamped against 255 and the bottom 8 bits used as the fog index. This should also take into account the number of depth bits there are.
9	InvertFI	✓	✓	x	This bit, when set, inverts the fog index before it is used to interpolates between the fragment's color and the fog color. This is usually 0 when fog values are used and 1 for Z values. Fog values are set up so they decrease with increasing depth and obviously Z values increase with increasing depth.
10…31	Unused	0	0	x	

Notes: The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

[18] Logic Op register readback is via the main register only

FogTable[0…15]
FogTable[16…31]
FogTable[32…47]
FogTable[48…63]

Name	Type	Offset	Format
FogTable[0..15]	Fog	0xB100…B178	Bitfield
FogTable[16…31]	Fog	0xB180…B1F8	Bitfield
FogTable[32…47]	Fog	0xB200…B278	Bitfield
FogTable[48…63]	Fog	0xB280…B2F8	Bitfield
	Control registers		

Bits	Name	Read	Write	Reset	Description
0…7		✓	✓	x	Fog index at tag +0
8…15		✓	✓	x	Fog index at tag +1
16…23		✓	✓	x	Fog index at tag +2
24…31		✓	✓	x	Fog index at tag +3

Notes: The fog index extracted from the DDA (either as a fog or z value as outlined above) can be used directly to control the blend, or it can be mapped via a table so some non-linear transfer function can be used.

The fog table is organised as 256 x 8 so the 8 bit input fog index is mapped to an 8 bit output fog index. The fog table is loaded by the FogTable0…FogTable63 registers and each holds 4 fog values at a time. FogTable0, byte 0 loads the mapping for fog index 0, byte 1 for fog index 1, etc..
The fog table is enabled by the Table bit in FogMode and is independent of how the initial fog index is generated

ForegroundColor

Name	Type	Offset	Format
ForegroundColor	LogicOps *Control register*	0xB0C0	Integer

Bits	Name	Read	Write	Reset	Description
0...31	Foreground Color	✓	✓	x	32 bit integer

Notes: With BackgroundColor, holds the foreground and background color values. The color format is in the raw framebuffer format and 8 or 16 bit pixels are automatically replicated to fill the 32 bits of register.

FStart

Name	Type	Offset	Format
FStart	Fog *Control register*	0x86A0	Fixed point

Bits	Name	Read	Write	Reset	Description
0…21	Fraction	✓	✓	x	
22…31	Integer	✓	✓	x	

Notes: Fog Coefficient start value. The value is in 2's complement 10.22 fixed point format.

GIDMode
GIDModeAnd
GIDModeOr

Name	**Type**	**Offset**	**Format**
GIDMode	Localbuffer	0xB538	Bitfield
GIDMode And	Localbuffer	0x B5B0	Bitfield Logic Mask
GIDMode Or	Localbuffer	0x B5B8	Bitfield Logic Mask
	Control registers		

Bits	Name	Read [19]	Write	Reset	Description
0	Fragment Enable	✓	✓	x	This bit, when set, causes GID testing to occur on fragments. If the test fails then the fragment is discarded
1	Span Enable	✓	✓	x	This bit, when set, allows the span pixel mask to be modified by GID testing each pixel. The mask is modified to disable those pixels which fail the test.
2…5	Compare Value	✓	✓	x	This field holds the 4 bit GID value to compare against. Unused bits (where the GID width in the local buffer format is less than 4 bits) should be set to zero.
6…7	Compare Mode	✓	✓	x	This field holds the comparison modes available for use during GID testing. The options are: 0 = Always pass 1 = Never pass (i.e. always fail) 2 = Pass when local buffer gid == CompareValue 3 = Pass when local buffer gid != CompareValue
8…9	Replace Mode	✓	✓	x	This field specifies the replacement mode. This is independent of the FragmentEnable bit (except when the replacement depends on the outcome of the GID test). The options are: 0 = Always replace 1 = Never replace 2 = Replace on GID test pass. 3 = Replace on GID test fails
10…13	Replace Value	✓	✓	x	This field holds the 4 bit GID value to replace the value read from the local buffer, if the replace mode is satisfied.
13…31	Reserved	0	0	x	Reserved

Notes: This register defines the Localbuffer GID operation.

The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

[19] Logic Op register readback is via the main register only

GlyphData

Name	Type	Offset	Format
GlyphData	2DSetup *Control register*	0xB660	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Packed data	✓	✓	x	Glyph data byte stream

Notes: A byte stream of glyph data (packed four to a word) can be downloaded and automatically chopped up and padded to the necessary width for the texture units to use as a bitmap. For example a gyph with a width between 17 and 24 pixels will be sent down as a stream of bytes and each triplet of bytes will be padded with zero and sent to be written into memory. If the input words have their bytes labelled:

First word: DCBA (A is the least significant byte)
Second word: HGFE

Then the output words send on to the rasterizer are:

First word: 0CBA
Second word: 0FED

GlyphPosition

Name	Type	Offset	Format
GlyphPosition	2DSetup *Control register*	0xB608	Integer

Bits	Name	Read	Write	Reset	Description
0…15	X offset	✓	✓	x	2's complement X coordinate
16…31	Y offset	✓	✓	x	2's complement Y coordinate

Notes: This register defines the glyph origin for use by the Render2DGlyph command. This register is updated by the Render2DGlyph command and the updated values will be read back or context dumped.

GStart

Name	Type	Offset	Format
GStart	Color *Control register*	0x8798	Fixed point number

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	Unused	0	0	x	

Notes: Used to set the initial Green value for a vertex when in Gouraud shading mode. The value is 24 bit 2's complement fixed point numbers in 9.15 format.

HeadPhysicalPageAllocation[0…3]

Name	Type	Offset	Format
HeadPhysicalPageAllocation [0…3]	Framebuffer *Control register*	0xB480	Integer

Bits	Name	Read	Write	Reset	Description
0…15	Address	✓	✓	x	16 bit integer value from 0 to 65535

Notes: These registers hold the head page for memory pools 0…3. This is usually the most recently referenced physical page in the pool of the working set. The range of physical pages is 0…65535

HostinDMAAddr

Name	Type	Offset	Format
DMAAddr	Input Control Register	0x8938	Bitfield

Bits	Name	Read	Write	Reset	Description
0…1	Reserved	0	0	x	
2…31	Address	✓	✓	x	Address

Notes: This register holds the byte address of the next DMA buffer to read from (reading doesn't start until the *DMACount* command). The bottom two bits of the address are ignored. This register should not be confused with the PCI register of the same name. *DMAAddr* must be loaded by itself and not as part of any increment, hold or indexed group. See also: *DMACount*.

HostinID

Name	Type	Offset	Format
HostinID	Delta *Control register*	0x8900	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Data	✓	✓	x	User-defined field

Notes: The HostInID register can be used to mark any point in the command stream so that the use of index and vertex buffers can be monitored. This register is loaded with an ID field; like the DMA address register, which can be read at any time.

HostInState

Name	Type	Offset	Format
HostInState	Delta *Control register*	0x8918	Integer

Bits	Name	Read	Write	Reset	Description
0…31	State data	✓	✓	x	32 bit value

Notes: This register is used to store a retained state that must be restored if a context switch occurs part way through a primitive.

HostInState2

Name	Type	Offset	Format
HostInState2	Delta *Control register*	0x8940	Integer

Bits	Name	Read	Write	Reset	Description
0…31	State data	✓	✓	x	32 bit value

Notes: This register is used to store a retained state that must be restored if a context switch occurs part way through a primitive.

IndexBaseAddress

Name	**Type**	**Offset**	**Format**
IndexBaseAddress	Input *Control register*	0xB700	Integer

Bits	Name	Read	Write	Reset	Description
0	Reserved	✓	✓	x	Reserved
1…16	Address	✓	✓	x	16 bit address of base of buffer

Notes:

IndexedDoubleVertex

Name	**Type**	**Offset**	**Format**
IndexedDoubleVertex	Input *Control register*	0xB7B0	Integer

Bits	Name	Read	Write	Reset	Description
0…15	Index0	×	✓	x	Offset into vertex buffer
16…31	Index1	×	✓	x	Offset into vertex buffer

Notes:

IndexedLineList

Name	Type	Offset	Format
IndexedLineList	Input *Control register*	0xB728	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Count	✕	✓	x	Number of indices in primitive

Notes:

IndexedLineStrip

Name	Type	Offset	Format
IndexedLineStrip	Input *Control register*	0xB730	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Count	✕	✓	x	Number of indices in primitive

Notes:

IndexedPointList

Name	Type	Offset	Format
IndexedPointList	Input *Control register*	0xB738	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Count	✕	✓	x	Number of indices in primitive

Notes:

IndexedPolygon

Name	Type	Offset	Format
IndexedPolygon	Input *Control register*	0xB740	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Count	✕	✓	x	Number of indices in primitive

Notes:

IndexedTriangleFan

Name	Type	Offset	Format
IndexedTriangleFan	Input *Control register*	0xB718	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Count	✕	✓	x	Number of indices in primitive

Notes:

IndexedTriangleList

Name	Type	Offset	Format
IndexedTriangleList	Input *Control register*	0xB710	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Count	✕	✓	x	Number of indices in primitive

Notes:

IndexedTriangleStrip

Name	**Type**	**Offset**	**Format**
IndexedTriangleStrip	Input *Control register*	0xB720	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Count	✗	✓	x	Number of indices in primitive

Notes:

IndexedVertex

Name	**Type**	**Offset**	**Format**
IndexedVertex	Input *Control register*	0xB7A8	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Index	✗	✓	x	Offset into index buffer

Notes:

InvalidateCache

Name	**Type**	**Offset**	**Format**
InvalidateCache	Texture *Command*	0xB358	Bitfield

Bits	Name	Read	Write	Reset	Description
0	Bank 0	✗	✓	x	Invalidate bank 0 of Primary Cache
1	Bank 1	✗	✓	x	Invalidate bank 1 of Primary Cache
2	TLB	✗	✓	x	Invalidate TLB
3…31	Unused	0	0	x	Reserved

Notes: This command invalidates the cache. The bottom three bits control what it to be invalidated.

KdBStart

Name	Type	Offset	Format
KdBStart	Texture *Control register*	0x8D30	Fixed point

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	reserved	0	0	x	

Notes: KdBStart holds the start value for the Blue Kd color component. The format is 24 bit 2's complement fixed point numbers in 9.15 format.

KdGStart

Name	Type	Offset	Format
KdGStart	Texture *Control register*	0x8D18	Fixed point

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	Unused	0	0	x	

Notes: *KdGStart* holds the start value for the Green Kd color component. The format is 24 bit 2's complement fixed point numbers in 9.15 format.

KdRStart

Name	Type	Offset	Format
KdRStart	Texture *Control register*	0x8D00	Fixed point

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	Unused	0	0	x	

Notes: KdRStart holds the start value for the Red Kd color component. The format is 24 bit 2's complement fixed point numbers in 9.15 format.

KdStart

Name	Type	Offset	Format
KdStart	Texture *Control register*	0x86E0	Fixed point

Bits	Name	Read	Write	Reset	Description
0…21	Fraction	✓	✓	x	
22…23	Integer	✓	✓	x	
24…31	Unused	0	0	x	

Notes: Initial values for Kd (diffuse). The value is 2.22 2's complement fixed point format.

KsBStart

Name	Type	Offset	Format
KsBStart	Texture *Control register*	0x8CB0	Fixed point

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	Unused	0	0	x	

Notes: KsBStart holds the start value for the Blue Ks color components. The format is 24 bit 2's complement fixed point numbers in 9.15 format.

KsGStart

Name	Type	Offset	Format
KsGStart	Texture *Control register*	0x8C98	Fixed point

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	reserved	0	0	x	

Notes: KsGStart holds the start value for the Green Ks color component. The format is 24 bit 2's complement fixed point numbers in 9.15 format.

KsRStart

Name	Type	Offset	Format
KsRStart	Texture *Control register*	0x8C80	Fixed point

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	Unused	0	0	x	

Notes: KsRStart holds the start values for the Red Ks color component. The format is 24 bit 2's complement fixed point numbers in 9.15 format.

LBClearDataL

Name	Type	Offset	Format
LBClearDataL	Localbuffer *Control register*	0xB550	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Address	✓	✓	x	32 bit integer value

Notes: This register holds the lower 32 bits of data to write into the local buffer (if so enabled) during a span operation. The data should be in the correct format to match up with the size and position of the depth, stencil and grapics ID fields.

LBClearDataU

Name	Type	Offset	Format
LBClearDataU	fer ***Control register***		

Bits	Name	Read	Write	Reset	Description
0…31	Address	✓	✓	x	32 bit integer value from 0 to 65535

Notes: This register holds the upper 8 bits of data to write into the local buffer (if so enabled) during a span operation. The data should be in the correct format to match up with the size and position of the depth, stencil, grapics ID and fast clear planes fields.

LBDepth

Name	Type	Offset	Format
LBDepth	Depth *Control register*	0x88B0	Integer

Bits	Name	Read	Write	Reset	Description
0…31	LBDepth	✗	✓	x	32 bit integer value

Notes: Internal register used in image upload of the depth buffer. This register should not be written to. It is documented here to give the tag value and format of the data which is read from the Host Out FIFO. Where the depth(Z) buffer width is less than 32bits, the depth value is right justified and zero extended.

LBDestReadBufferAddr

Name	Type	Offset	Format
LBDestReadBufferAddr	Local buffer *Control register*	0xB510	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Address	✓	✓	x	32 bit value

Notes: This register holds the 32 bit base address of the source buffer in memory. The address is a byte address and should be aligned to the natural boundary for the selected local buffer pixel size.

LBDestReadBufferOffset

Name	Type	Offset	Format
LBDestReadBufferOffset	Localbuffer *Control register*	0xB518	Integer

Bits	Name	Read	Write	Reset	Description
0…15	X offset	✓	✓	x	2’s complement X offset
16…31	Y offset	✓	✓	x	2’s complement Y offset

Notes: These registers hold the offset added to the fragment's coordinate for each destination buffer. The new coordinate is used for address calculations. This offset allows, for example, window relative coordinates to be converted into screen relative ones prior to patching (patching only works screen relative).

LBDestReadEnables
LBDestReadEnablesAnd
LBDestReadEnablesOr

Name	Type	Offset	Format
LBDestReadEnables	Localbuffer	0xB508	Bitfield
LBDestReadEnablesAnd	Localbuffer	0xB590	Bitfield Logic Mask
LBDestReadEnablesOr	Localbuffer	0xB598	Bitfield Logic Mask
	Control registers		

Bits	Name	Read [20]	Write	Reset	Description
0…3	E1 to E3	✓	✓	x	These bits are the Enable bits. Software assigns these to major modes which can be enabled or disabled (such as Depth Testing) it wants the LB Read Unit to track so destination reads are automatically done when necessary. When a bit is 1 it is enabled. E0…E3 are used for fragments.
4…7	E4 to E7	✓	✓	x	Used for spans
8…11	R0 to R3	✓	✓	x	These are Read bits. Software assigns these to operations within a major mode which require reads. For example the major mode would be Depth Testing, but not all depth test option require the destination buffer to be read. When a bit is 1 a read is required. R0…R3 are used for fragments.
12…15	R4 to R7	✓	✓	x	Used for spans
24…31	Reserved	0	0	x	Reserved

Notes: This new register contains 8 pairs of bits which the software can assign to activities which could require local buffer reads. The pairs of bits comprise an E bit and a R bit. The E bit reflects a major mode enable (e.g. stencil) and is set whenever that mode is enabled. The R bit is set when the operation within the major mode requires a read.

For example:

E0 = Depth Enable R0 = Set whenever a depth mode requires a read
E1 = Stencil Enable R1 = Set whenever a stencil operation requires a read
E2 = GID enable R2 = Set whenever the GID testing is required.

The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

[20] Logic Op register readback is via the main register only

LBDestReadMode
LBDestReadModeAnd
LBDestReadModeOr

Name	**Type**	**Offset**	**Format**
LBDestReadMode	Localbuffer	0xB500	Bitfield
LBDestReadModeAnd	Localbuffer	0xB580	Bitfield Logic Mask
LBDestReadModeOr	Localbuffer	0xB588	Bitfield Logic Mask
	Control registers		

Bits	Name	Read[21]	Write	Reset	Description
0	Enable	✓	✓	x	This bit, when set, causes fragments or spans to read from the destination buffer
1	Reserved	×	×	x	
2…4	StripePitch	✓	✓	x	This field specifies the number of scanlines between the first scanline in a stripe and the first scanline in the next stripe. (It would normally be set to a number of RXs * StripeHeight). The options are: 0 = 1 1 = 2 2 = 4 3 = 8 4 = 16 5 = 32 6 = 64 7 = 128 This field will normally be set to zero for P3.
5…7	StripeHeight	✓	✓	x	This field specifies the number of scanlines in a stripe. The options are: 0 = 1 1 = 2 2 = 4 3 = 8 4 = 16 This field will normally be set to zero for P3.
8	Layout	✓	✓	x	This field selects the layout of the pixel data in memory for the destination buffer. The options are: 0 = Linear 1 = Patch64
9	Origin	✓	✓	x	This field selects where the window origin is for the destination buffer. The options are: 0 = Top Left. 1 = Bottom Left
10	UseRead Enables	✓	✓	x	When this bits is set the enables in the LBDestReadEnables register are used to determine if a destination read is required. The Enable bit must also be set as well for a read to occur.
11	Packed16	✓	✓	x	When this bit is set the pixel size is 16 bits so a single memory word can hold 8 depht values.
12…23	Width	✓	✓	x	This field holds the width of the destination buffer. Its range is 0…4095.

Notes: Defines the localbuffer destination read operation. The destination address calculations are controlled by the *LBDestReadMode* register and the address is a function of X, Y, *LBDestReadBufferAddr*, *LBDestReadBufferOffset*, width and Packed16 parameters.

The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

[21] Logic Op register readback is via the main register only

LBReadFormat

Name	Type	Offset	Format
LBReadFormat	Localbuffer *Control register*	0x8888	Bitfield

Bits	Name	Read	Write	Reset	Description
0…1	DepthWidth	✓	✓	x	This field specifies the width of the depth field. The depth field always starts at bit position 0. The width options are: 0 = 16 bits 1 = 24 bits 2 = 31 bits 3 = 15 bits When the depth width is 15 the GID and Stencil fields are ignored and a one bit GID and Stencil are taken from bit 15. Only one of the GID or Stencil operation are enabled to select the desired field type.
2…5	StencilWidth	✓	✓	x	This field specifies the width of the stencil field. The legal range of values are 0…8. The stencil field always starts at bit position given in the next field.
6…10	StencilPosition	✓	✓	x	This field holds position of the least significant bit of the stencil field. The legal range of values are 0…23, representing bit positions 16…39 respectively.
11…14	FCPWidth	0	0	x	Reserved
15…19	FCPPosition	0	0	x	Reserved
20…22	GIDWidth	✓	✓	x	This field specifies the width of the Graphics ID field. The legal range of values are 0…4. The GID field always starts at bit position given in the next field.
23…27	GIDPosition	✓	✓	x	This field holds position of the least significant bit of the Graphics ID field. The legal range of values are 0…23, representing bit positions 16…39 respectively.
28…31	Unused	0	0	x	

Notes: This register defines the position and width of the depth, stencil and GID (Graphics ID) in the data read back from the local buffer.

Notes: LB ReadFormat register definition has changed to allow more flexible sizing and positioning of the GID and stencil fields.

FCP is not supported on Permedia3 - the fields are reserved for future use.

LBSourceReadBufferAddr

Name	**Type**	**Offset**	**Format**
LBSourceReadBufferAddr	Localbuffer *Control register*	0xB528	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Address	✓	✓	x	32 bit value

Notes: This register holds the 32 bit base address of the source buffer in memory. The address is a byte address and should be aligned to the natural boundary for the selected pixel size.

LBSourceReadBufferOffset

Name	**Type**	**Offset**	**Format**
LBSourceReadBufferOffset	Localbuffer *Control register*	0xB530	Integer

Bits	Name	Read	Write	Reset	Description
0…15	X offset	✓	✓	x	2’s complement X offset
16…31	Y offset	✓	✓	x	2’s complement Y offset

Notes: This register hold the offset added to the fragment's coordinate for the source buffer. The new coordinate is used for address calculations. This offset allows, for example, window relative coordinates to be converted into screen relative ones prior to patching (patching only works screen relative).

LBSourceReadMode
LBSourceReadModeAnd
LBSourceReadModeOr

Name	Type	Offset	Format
LBSourceReadMode	Alpha Blend	0xB520	Bitfield
LBSourceReadModeAnd	Alpha Blend	0xB5A0	Bitfield Logic Mask
LBSourceReadModeOr	Alpha Blend	0xB5A8	Bitfield Logic Mask
	Control registers		

Bits	Name	Read [22]	Write	Reset	Description
0	Enable	✓	✓	x	This bit, when set, causes fragments to be read from the source buffer. If this bit is clear then no source reads are made.
1	Reserved	0	0	x	
2…4	StripePitch	✓	✓	x	This field specifies the number of scanlines between the first scanline in a stripe and the first scanline in the next stripe. It would normally be set to number of RXs * StripeHeight. The options are: 0 = 1 4 = 16 1 = 2 5 = 32 2 = 4 6 = 64 3 = 8 7 = 128 This field will normally be set to zero for P3.
5…7	StripeHeight	✓	✓	x	This field specifies the number of scanlines in a stripe. The options are: 0 = 1 3 = 8 1 = 2 4 = 16 2 = 4 This field will normally be set to zero for P3.
8	Layout	✓	✓	x	This field selects the layout of the pixel data in memory for the source buffer. The options are: 0 = Linear 1 = Patch64
9	Origin	✓	✓	x	This field selects where the window origin is. The options are: 0 = Top Left. 1 = Bottom Left
10	Packed16	✓	✓	x	When this bit is set the pixel size is 16 bits so a single memory word can hold 8 depth values.
11…22	Width	✓	✓	x	This field holds the width of the destination buffer. Its range is 0…4095.
23…31	Reserved	0	0	x	

Notes: This register defines the Localbuffer source read operation. The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

[22] Logic Op register readback is via the main register only

LBStencil

Name	Type	Offset	Format
LBStencil	Localbuffer *Command*	0x88A8	Bitfield

Bits	Name	Read	Write	Reset	Description
0…7	Stencil	✗	✗	x	
8…15	Reserved	✗	✗	x	
16…19	GID	✗	✗	x	
20…31	Reserved	0	0	x	

Notes: Internal register used in upload of the stencil buffer. It should not be written to and is documented here only to give the tag value and format of the data when read from the host out FIFO.

LBWriteBufferAddr

Name	Type	Offset	Format
LBWriteBufferAddr	Localbuffer *Control register*	0xB540	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Address	✓	✓	x	32 bit value

Notes: This register holds the 32 bit base address of the source buffer in memory. The address is a byte address and should be aligned to the natural boundary for the selected pixel size.

LBWriteBufferOffset

Name	Type	Offset	Format
LBWriteBufferOffset	Localbuffer *Control register*	0xB548	Integer

Bits	Name	Read	Write	Reset	Description
0…15	X offset	✓	✓	x	2’s complement X offset
16…31	Y offset	✓	✓	x	2’s complement Y offset

Notes: This register holds the offset added to the fragment's coordinate for the destination buffer. The new coordinate is used for address calculations. This offset allows, for example, window relative coordinates to be converted into screen relative ones prior to patching (patching only works screen relative).

LBWriteFormat

Name	Type	Offset	Format
LBWriteFormat	Localbuffer *Control register*	0x88C8	Bitfield

Bits	Name	Read	Write	Reset	Description
0…1	DepthWidth	✓	✓	x	This field specifies the width of the depth field. The depth field always starts at bit position 0. The width options are: 0 = 16 bits 1 = 24 bits 2 = 31 bits 3 = 15 bits When the depth width is 15 the GID and Stencil fields are ignored and a one bit GID and Stencil are taken from bit 15. Only one of the GID or Stencil operation are enabled to select the desired field type.
2…5	StencilWidth	✓	✓	x	This field specifies the width of the stencil field. The legal range of values are 0…8. The stencil field always starts at bit position given in the next field.
6…10	StencilPosition	✓	✓	x	This field holds position of the least significant bit of the stencil field. The legal range of values are 0…23, representing bit positions 16…39 respectively.
11…19	Reserved	0	0	x	
20…22	GIDWidth	✓	✓	x	This field specifies the width of the Graphics ID field. The legal range of values are 0…4. The GID field always starts at bit position given in the next field.
23…27	GIDPosition	✓	✓	x	This field holds position of the least significant bit of the Graphics ID field. The legal range of values are 0…23, representing bit positions 16…39 respectively.
28…31	Reserved	0	0	x	

Notes: This register defines the position and width of the depth, stencil, GID (Graphics ID) in the data read back from the local buffer.

LBWriteMode
LBWriteModeAnd
LBWriteModeOr

Name	Type	Offset	Format
LBWriteMode	Localbuffer	0x88C0	Bitfield
LBWriteModeAnd	Localbuffer	0xAC80	Bitfield
LBWriteModeOr	Localbuffer	0xAC88	Bitfield
	Control register		

Bits	Name	Read [23]	Write	Reset	Description
0	WriteEnable	✓	✓	x	This bit, when set, causes fragments or spans to written to the destination buffer. Note each byte must also be enabled in the ByteEnables field.
1…2	Reserved	0	0	x	
3…5	StripePitch	✓	✓	x	This field specifies the number of scanlines between the first scanline in a stripe and the first scanline in the next stripe. It would normally be set to number of RXs * StripeHeight. The options are: 0 = 1 4 = 16 1 = 2 5 = 32 2 = 4 6 = 64 3 = 8 7 = 128 This field will normally be set to zero for P3.
6…8	StripeHeight	✓	✓	x	This field specifies the number of scanlines in a stripe. The options are: 0 = 1 3 = 8 1 = 2 4 = 16 2 = 4 This field will normally be set to zero for P3.
9	Layout	✓	✓	x	This field selects the layout of the pixel data in memory for the destination buffer. The options are: 0 = Linear 1 = Patch64
10	Origin	✓	✓	x	This field selects where the window origin is for the destination buffer. The options are: 0 = Top Left. 1 = Bottom Left
11	Packed16	✓	✓	x	When this bit is set the pixel size is 16 bits so a single memory word can hold 8 depth values.
12…23	Width	✓	✓	x	This field holds the width of the destination buffer. Its range is 0…4095.

[23] Logic Op register readback is via the main register only

24…28	ByteEnables	✓	✓	x	This field holds the byte enables for each byte in the pixel. A byte enable bit must be set for the corresponding byte to be written. Ideally the depth, stencil, etc. fields are byte aligned and integral bytes in length so these can be used to disable modifying a field, otherwise read-modify-write operations will need to be done.
29…31	Operation	✓	✓	x	This field defines where the data is to be taken from to do the write and what is to happen to it afterwards. This is only of interest during an upload or download operation. The options are: 0 = No operation 1 = Download depth 2 = Download stencil 3 = Upload depth 4 = Upload stencil

Notes: The write requests have two forms:

- Single pixel. This is the normal mode for 3D operation but is only used for exotic 2D operations. The calculated address is always a pixel address and this is shifted to take into account the width of a pixel (16 or 32 bits) in calculating the memory address and byte enables. The pixel data (Z, stencil and GID) are formatted and shifted into the correct byte lanes for the memory.
- Pixel spans. Spans are useful for clearing down the local buffer but do not use any block fill capabilities of the memory (these are only available through the FB Write Unit), although 4 or 8 pixels will be cleared down per cycle.
- N.B Write operation is not compatible with GLINT MX for programming purposes.

The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

LineCoord0

Name	Type	Offset	Format
LineCoord0	Delta *Command*	0x9760	Bitfield

Bits	Name	Read	Write	Reset	Description
0..15	X	✗	✓	x	2's complement X
16..31	Y	✗	✓	x	2's complement Y

Notes:
- *LineCoord0* loads vertex store 0
- *LineCoord1* loads vertex store 1.
- *DrawLine0* draws a line from vertex 0 to vertex1
- *DrawLine1* draws a line from vertex 1 to vertex 0.

Note: to confirm LineCoord tages have written values correctly, readback using *V0FloatX*, *V0FloatY* and similar registers..

LineCoord1

Name	Type	Offset	Format
LineCoord1	Delta *Command*	0x9770	Bitfield

Bits	Name	Read	Write	Reset	Description
0..15	X	✗	✓	x	2's complement X
16..31	Y	✗	✓	x	2's complement Y

Notes:
- *LineCoord0* loads vertex store 0
- *LineCoord1* loads vertex store 1.
- *DrawLine0* draws a line from vertex 0 to vertex1
- *DrawLine1* draws a line from vertex 1 to vertex 0.

Note: to confirm LineCoord tages have written values correctly, readback using *V0FloatX*, *V0FloatY* and similar registers.

LineStippleMode
LineStippleModeAnd
LineStippleModeOr

Name	Type	Offset	Format
LineStippleMode	Stipple	0x81A8	Bitfield
LineStippleModeAnd	Stipple	0xABC0	Bitfield Logic Mask
LineStippleModeOr	Stipple	0xABC8	Bitfield Logic Mask
	Control register		

Bits	Name	Read	Write	Reset	Description
0	StippleEnable	✓	✓	x	This field, when set, enables the stippling of lines. The LineStippleEnable bit in the *Render* command must also be set.
1…9	RepeatFactor	✓	✓	x	This field holds the positive repeat factor for stippled lines. The repeat factor stored here is one less than the desired repeat factor.
10…25	StippleMask	✓	✓	x	This field holds the stipple pattern.
26	Mirror	✓	✓	x	This field, when set, will mirror the StippleMask before it is used.
27…31	Unused	0	0	x	

Notes: Controls line stippling:

- The repeat factor is set to one less than the required value.
- The least significant bit of the *UpdateLineStippleCounters* register, controls loading the line stipple counters - if set the line stipple counters are loaded with the previously saved values. If reset, the counters are cleared to zero.
- The counters can also be reset by means of the ResetLineStipple bit in the Render command.
- The Enable bit in the *LineStippleMode* register is qualified by the LineStippleEnable bit in the *Render* Command.

LoadLineStippleCounters

Name	Type	Offset	Format
LoadLineStippleCounters	Global *Command*	0x81B0	Bitfield

Bits	Name	Read	Write	Reset	Description
0…3	LiveBit Counter	✗	✓	x	
4…12	LiveRepeat Counter	✗	✓	x	
13…16	SegmentBit Counter	✗	✓	x	
17…25	SegmentRepeat Counter	✗	✓	x	
26…31	Unused	0	0	x	

Notes: Command used to restore the line stipple counters and segment register after a task switch. The counters are incremented during a line stipple so the value read from them, via the readback path may not match the value loaded in to them using this register.

LOD

Name	Type	Offset	Format
LOD	Texture *Control register*	0x83D0	Fixed point

Bits	Name	Read	Write	Reset	Description
0…7	Fraction	✓	✓	x	
8…11	Integer	✓	✓	x	
12…31	Reserved	0	0	x	Reserved for future use. Mask to 0.

Notes: Holds the computed level of detail value for texture 0. The format is 4.8 unsigned fixed point.

The Level Of Detail (LOD) calculates the approximate area a fragment projects onto the texture map. The LOD calculation is enabled by the EnableLOD bit in the TextureCoordMode register. When this bit is clear no LOD is calculated and a constant LOD from the LOD register is used (when it is required by the *TextureReadMode* register setting). The format is unsigned 4.8 fixed point and can be interpreted as follows:

- the integer part selects the higher resolution map of the pair to use with 0 using the map at the address given by TextureBaseAddress[0] register
- the fraction gives the between map interpolation coefficient measured from the higher resolution map selected.

LOD1

Name	Type	Offset	Format
LOD1	Texture *Control register*	0x8448	Fixed point

Bits	Name	Read	Write	Reset	Description
0…7	Fraction	✓	✓	x	
8…11	Integer	✓	✓	x	
12…31	Reserved	0	0	x	

Notes: Holds the constant level of detail to use for mip mapping from texture 1. The format is 4.8 unsigned fixed point.

The Level Of Detail (LOD) calculates the approximate area a fragment projects onto the texture map. The LOD calculation is enabled by the EnableLOD bit in the TextureCoordMode register. When this bit is clear no LOD is calculated and a constant LOD from the LOD register is used (when it is required by the *TextureReadMode* register). The format is unsigned 4.8 fixed point and can be interpreted as follows:

- the integer part selects the higher resolution map of the pair to use with 0 using the map at the address given by TextureBaseAddress[0] register
- the fraction gives the between map interpolation coefficient measured from the higher resolution map selected.

LODRange0

Name	Type	Offset	Format
LODRange0	Texture *Control register*	0xB348	Fixed point

Bits	Name	Read	Write	Reset	Description
0…11	Min	✓	✓	x	2’s complement 4.8 fixed point fraction
12…23	Max	✓	✓	x	2’s complement 4.8 fixed point integer
24…31	Reserved	0	0	x	

Notes: This register holds the clamping range for lod0 calculations. Bits 0-11 define the minimum value, bits 12-23 hold the maximum value.

LODRange1

Name	Type	Offset	Format
LODRange1	Texture *Control register*	0xB350	Fixed point

Bits	Name	Read	Write	Reset	Description
0…11	Min	✓	✓	x	2’s complement 4.8 fixed point fraction
12…23	Max	✓	✓	x	2’s complement 4.8 fixed point integer
24…31	Reserved	0	0	x	

Notes: This register holds the clamping range for lod1 calculations. Bits 0-11 define the minimum value, bits 12-23 hold the maximum value.

LogicalOpMode
LogicalOpModeAnd
LogicalOpModeOr

Name	**Type**	**Offset**	**Format**
LogicalOpMode	Logic Ops	0x8828	Bitfield
LogicalOpModeAnd	Logic Ops	0xAEC0	Bitfield Logic Mask
LogicalOpModeOr	Logic Ops	0xAEC8	Bitfield Logic Mask
	Control registers		

Bits	Name	Read [24]	Write	Reset	Description
0	Enable	✓	✓	x	When set causes the fragment's color to be logial op'ed under control of the remaining bits in this register. When clear the fragment color remains unchanged (but may later to effected by write masking).
1…4	LogicOp	✓	✓	x	This field defines the logical op function to use. The options are: 0 = Clear (0) 1 = And(S & D) 2 = AndReverse (S & ~D) 3 = Copy (S) 4 = AndInvert (~S & D) 5 = Noop (D) 6 = Xor (S ^ D) 7 = Or (S | D) 8 = Nor (~(S | D); 9 = Equiv (~(S ^ D); 10 = Invert (~D) 11 = OrReverse (S | ~D) 12 = CopyInvert (~S) 13 = OrInvert (~S | D) 14 = Nand (~(S & D); 15 = Set (1) where: S is Color or FBSourceData D is FBData
5	UseConstantFB WriteData	✓	✓	x	There is no longer any performance advantage to using this bit but it is retained for backwards compatability.
6	BackgroundEn able	✓	✓	x	This bit, when set, enables a different logical operation to be done for background pixels. If this bit is clear then the same logical operation is applied to foreground and background pixels. Setting this bit when the Enable field is zero has no effect. A background pixel is a pixel whose corresponding bit in the color mask is zero.
7…10	BackgroundLog icalOp	✓	✓	x	This field specifies the logical operation to apply to background pixels, if this has been enabled by the BackgroundEnable field. The options and field values are the same as the LogicalOp field.

[24] Logic Op register readback is via the main register only

11	UseConstantSource	✓	✓	x	This field, when set, causes the source data to be taken from the ForegroundColor register, otherwise it is taken from the fragment, if needed. The color format is in the raw framebuffer format and 8 or 16 bit pixels should have their color replicated to fill the full 32 bits.
12	OpaqueSpan	✓	✓	x	This bit determines how constant colour spans are to be processed. The two options are: 0 = Transparent 1 = Opaque Transparent spans take the source pixel colour from the message stream or the **ForegroundColour** register as appropriate. Opaque spans take the source pixel colour from the message stream or register. The **ForegroundColour** register is used when the corresponding bit in the SpanColourMask is 1, otherwise the **BackgroundColour** register is used.
12…31	Unused	0	0	x	

Notes: The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

LogicalTexturePageTableAddr

Name	Type	Offset	Format
LogicalTexturePageTableAddr	Texture	0xB4D0	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…31	Address	✓	✓	x	32 bit value

Notes: This register holds the base address of the Logical Texture Page Table. The address should be aligned to a 64 bit boundary.

LogicalTexturePageTableLength

Name	Type	Offset	Format
LogicalTexturePageTableLength	Texture	0xB4D8	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…16	Logical page count	✓	✓	x	17 bit integer value from 0 to 65536

Notes:	This register holds the number of logical pages to be managed. Any logical pages past this value are folded to logical page 0. Setting this register to zero effectively disables logical to physical mapping. The legal range of values is 0…65536.

LUT[0…15]

Name	**Type**	**Offset**	**Format**
LUT[0..15]	LUT *Control registers*	0x8E80	Bitfield

Bits	Name	Read	Write	Reset	Description
0…7	Red	✓	✓	x	
8…15	Green	✓	✓	x	
16…23	Blue	✓	✓	x	
24…31	Alpha	✓	✓	x	

Notes: These registers allow the lower 16 entries of the LUT to be loaded and read back directly.

LUTAddress

Name	**Type**	**Offset**	**Format**
LUTAddress	Texture *Control register*	0x84D0	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Address	✓	✓	x	32 bit value

Notes: This register holds the physical address of a block of data to load into the LUT from memory. This is given as a byte address, but the bottom 4 bits are ignored so the address is effectively aligned to a 128 bit memory word.

LUTData

Name	Type	Offset	Format
LUTData	Texture *Control register*	0x84C8	Integer

Bits	Name	Read	Write	Reset	Description
0…31	LUT data word	✓	✓	x	32 bit value

Notes: This register holds the 32 bits of data to load into the LUT. The data can be loaded in 'as is', have its red and green components swapped over or converted into a replicated 16 bit format.

LUT readback is done by first reading the *LUTIndex* register. As well as returning the current LUT index it has the additional effect of setting the ReadIndex counter to zero. The ReadIndex counter is only used during readback and is not the same as the LUTIndex used for loading the LUT via the message stream. Each subsequent read from the *LUTData* register returns the LUT data at the ReadIndex and the ReadIndex counter is incremented. The ReadIndex counter wraps from 255 to 0.

LUTIndex

Name	Type	Offset	Format
LUTIndex	Texture *Control register*	0x84C0	Integer

Bits	Name	Read	Write	Reset	Description
0…7	Index	✓	✓	x	8 bit integer value from 0 to 255
8…31	Unused	0	0	x	

Notes: This register holds the start index to update the LUT at when LUT data message is written. The index is automatically incremented after each load and wraps from 255 to 0. Readback from LUTIndex has side effect of clearing the *ReadIndex* register.

LUTMode
LUTModeAnd
LUTModeOr

Name	Type	Offset	Format
LUTMode	LUT	0xB378	Bitfield
LUTModeAnd	LUT	0xAD70	Bitfield Logic Mask
LUTModeOr	LUT	0xAD78	Bitfield Logic Mask
	Control registers		

Bits	Name	Read [25]	Write	Reset	Description
0	Enable	✓	✓	x	When set causes the fragment or span data to be modified under control of the remaining bits in this register.
1	InColorOrder	✓	✓	x	This bit, when set, swaps the red and green bytes (i.e. bytes 0 and 2) of the 32 bit load data. This can be used to convert ARGB input data into ABGR data to match the internal processing format.
2…3	LoadFormat	✓	✓	x	This field controls how the 32 bit data is to be loaded into the LUT. The options are: 0 = Copy (i.e. no formatting). 1 = 565 Replicated 2 = 5551 Replicated The conversion from 8 bits to 1, 5 or 6 bits is done by subtracting half and truncating. The 16 bit value is replicated into both halves of the LUT.
4	LoadColorOrder	✓	✓	x	This bit controls the order the 16 bit color components are assembled in after the conversion while loading. The options are: 0 = BGR or ABGR 1 = RGB or ARGB

[25] Logic Op register readback is via the main register only

5…7	FragmentOpera tion	✓	✓	x	This field specifies the operation to be done on each fragment when not using spans to do the rendering. The options are: 0 = None 1 = IndexedTexture. The 8 bit indexed texels are converted into 32 bit true color values. 2 = Translate8To32. The fragment's red channel is converted into a 32 bit ABGR value using the LUT. 3 = Translate32To32. Each of the four color components are translated using its own LUT. 4 = MotionComp. The LUT holds motion compensation data held in Planar 411 format as 8 bit or 9 bit YUV values. This is indexed based on the fragments coordinates and expanded to 9 bits, if necessary, and assigned to the fragment's color. 5 = Pattern. The LUT holds an 8x8 pattern for the chosen pixel size and this is used to set the fragment's color . Note the SwapSD bit in the AlphaBlendColorMode register may need to be set if the pixel size is 8 or 16 bits.
8…10	SpanOperation	✓	✓	x	This field specifies the operation to be done on each pixel in a span. The options are: 0 = None 1 = SpanPattern. The LUT holds an 8x8 pattern for the chosen pixel size and this is used to set the block color or the span pixel data depending on the span operation bit in the *Render* command (constant color uses block color, variable color uses span pixel data). 2 = Translate8To8. Each byte is translated using its corresponding LUT channel (so 8 bytes can be translated in parallel). Normally the LUT is set up so all four byte channels hold the same data. 3 = Translate8To16. Each byte is translated using a pair of LUT channels to generate a 16 bit pixel. The LUT is set up so that pairs of channels hold the same data. This can be arranged automatically when the LUT is first loaded.. 4 = Translate8To32. Each byte is translated into a 32 bit pixel using the LUT. 5 = Translate32To32. Each byte is translated using its corresponding LUT channel (so 8 bytes can be translated in parallel). Normally the LUT is set up so all four byte channels hold different data.
11	MotionComp8 Bits	✓	✓	x	This bit, if set, specifies that the YUV data is held as 8 bit values, packed 4 per 32 bit LUT entry. If this bit is not set the YUV data is held as 9 bit values packed 2 per 32 bit LUT entry (on 16 bit boundaries within the 32 bit word).

12…14	XOffset	✓	✓	x	This field holds the X offset into the selected 8x8 pattern. This is used (together with the pixels X coordinate) to rotate the selects row of the pattern to give some control on its registration to the underlying rectangle.
15…17	YOffset	✓	✓	x	This field holds the Y offset into the selected 8x8 pattern. This is used (together with the pixels Y coordinate) to select which row of the pattern to use. This gives some control of the patterns registration to the underlying rectangle.
18…25	PatternBase	✓	✓	x	This field holds the base address of the pattern to use. There are no restrictions on where a pattern starts, other than it must start on a 32 bit boundary (i.e. the start cannot be part way through a LUT entry).
26	SpanCCXAlignment	✓	✓	x	This bit controls how the pattern is aligned along the X axis when Constant Color spans are used. The two options are: 0 = The first pixel in the span is taken from the pixel indexed for this row by XOffset. This is the normal method and fixes the pattern with respect to the screen (recall the block color registers are memory aligned). This preserves a vertical line in the pattern when applying to a trapezoid. 1 = The first pixel in the span is taken from (X + XOffset) % 8
27	SpanVCXAlignment	✓	✓	x	This bit controls how the pattern is aligned along the X axis when Constant Color spans are used. The two options are: 0 = The first pixel in the span is taken from the pixel indexed for this row by XOffset. 1 = The first pixel in the span is taken from (X + XOffset) % 8. This is the normal method and fixes the pattern with respect to the screen (recall these are done via normal writes so are not memory aligned). This preserves a vertical line in the pattern when applying to a trapezoid.

Notes: The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

LUTTransfer

Name	Type	Offset	Format
LUTTransfer	Texture *Command*	0x84D8	Bitfield

Bits	Name	Read	Write	Reset	Description
0…7	Start index	✓	✓	x	Index
8…14	Count	✓	✓	x	Count in 128 bit words.
15…31	Reserved	0	0	x	

Notes: This register initiates the transfer of data from memory into the LUT.

MaxHitRegion

Name	Type	Offset	Format
MaxHitRegion	Output *Command*	0x8C30	Bitfield

Bits	Name	Read	Write	Reset	Description
0…15	Maximum X	✗	✓	x	maximum X in 2's complement format.
16…31	Maximum Y	✗	✓	x	maximum Y in 2's complement format.

Notes: This register causes the current value of the *maxRegion* register to be written to the output FIFO under control of the *FilterMode* register (which may cull the data depending on the setting of the Statistics bits). The data field (on input) is not used.

MaxRegion

Name	Type	Offset	Format
MaxRegion	Output *Control register*	0x8C18	Bitfield

Bits	Name	Read	Write	Reset	Description
0…15	Maximum X	✗	✓	x	maximum X in 2's complement format.
16…31	Maximum Y	✗	✓	x	maximum Y in 2's complement format.

Notes: This register initialises the maximum region register. The register is updated during extent testing:

- During Picking it contains the max X,Y value for the Pick region.
- During Extent collection it is set to the initial minimum extent and is updated whenever a fragment with a higher X or Y value is generated, to reflect the new X or Y.

The *StatisticMode* register allows either fragments or those that were culled after being rasterised to be set as Eligible to update this register. Since register contents are updated during rendering it may not return the value previously written to it.

MinHitRegion

Name	Type	Offset	Format
MinHitRegion	Output *Control register*	0x8C28	Bitfield

Bits	Name	Read	Write	Reset	Description
0…15	Minimum X	✗	✓	x	minimum X in 2's complement format.
16…31	Minimum Y	✗	✓	x	minimum Y in 2's complement format.

Notes: This register causes the current value of the *minRegion* register to be written to the output FIFO under control of the *FilterMode* register (which may cull the data depending on the setting of the Statistics bits). The data field (on input) is not used.

MinRegion

Name	Type	Offset	Format
MinRegion	Output *Control register*	0x8C10	Bitfield

Bits	Name	Read	Write	Reset	Description
0…15	Minimum X	✗	✓	x	minimum X in 2's complement format.
16…31	Minimum Y	✗	✓	x	minimum Y in 2's complement format.

Notes: This register initialises the minimum region register. The register is updated during extent testing:

- During Picking it contains the max X,Y value for the Pick region.
- During Extent collection it is set to the initial minimum extent and is updated whenever a fragment with a higher X or Y value is generated, to reflect the new X or Y.

The *StatisticMode* register allows either active fragments or those that were culled after being rasterised to be set as Eligible to update this register. Since register contents are updated during rendering it may not return the value previously written to it.

Packed16Pixels

Name	**Type**	**Offset**	**Format**
Packed16Pixels	2DSetup *Command*	0xB638	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Data word	✗	✓	x	

Notes: Packed Downloads: The target register for the expanded pixel data is set up with the *DownloadTarget* command. Four bit packed pixel downloads are converted into eight bit packed pixels. The 8 and 16 packed pixels are particularly useful when downloading textures because spans (which take packed data) cannot be used when the target buffer layout is Patch2 or Patch32_2.

Each *Packed16Pixels* command will be expanded into 2 writes to the target register. If the input bytes are labelled DCBA (with byte A in bit positions 0…7) then this is converted to:

First word: 00BA (0 is the byte set to zero)
Second word: 000DC

Packed4Pixels

Name	**Type**	**Offset**	**Format**
Packed16Pixels	2DSetup *Command*	0xB668	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Data word	✗	✓	x	

Notes: Packed Downloads: The target register for the expanded pixel data is set up with the *DownloadTarget* command. Four bit packed pixel downloads are converted into eight bit packed pixels.

This register holds the packed nibble pixel data to expand out into packed byte pixel data. Each Packed4Pixels command will be expanded into two writes to the target register. If the input nibbles are labelled HGFEDCBA (with nibble A in bit positions 0…3) then this is converted to:

First word: 0C0D0A0B (0 is the nibble set to zero)
Second word: 0G0H0E0F

Packed8Pixels

Name	Type	Offset	Format
Packed8Pixels	2DSetup *Command*	0xB630	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Data word	✗	✓	x	

Notes: Packed Downloads: The target register for the expanded pixel data is set up with the *DownloadTarget* command.

This register holds the packed 8 bit pixel data to expand out into 4 seperate 8 bit pixels during the download. The data is sent to the register defined in DownloadTarget. Each Packed8Pixels command will be expanded into four writes to the target register. If the input bytes are labelled DCBA (with byte A in bit positions 0…7) then this is converted to:

First word:	000A	(0 is the byte set to zero)
Second word:	000B	
Third word:	000C	
Fourth word:	000D	

PhysicalPageAllocationTableAddr

Name	Type	Offset	Format
PhysicalPageAllocation TableAddr	Texture *Control register*	0xB4C0	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Address	✓	✓	x	32 bit value

Notes: This register holds the base address of the Physical Page Allocation Table. The address should be aligned to a 64 bit boundary.

PickResult

Name	**Type**	**Offset**	**Format**
PickResult	Output *Command*	0x8C38	Bitfield

Bits	Name	Read	Write	Reset	Description
0	Pick result	✕	✓	x	Flag
1…31	Reserved	✕	0	x	

Notes: This command causes the current value of the pick result flag to be written to the output FIFO under control of the FilterMode settings. The data field (on input) is not used.
Output = 0 for false or 1 for true.

PixelSize

Name	Type	Offset	Format
PixelSize	Rasterizer *Command*	0x80C0	Bitfield

Bits	Name	Read	Write	Reset	Description
0…1	Global	✓	✓	x	All units, if bit 31 is zero, otherwise
2…3	Rasterizer	✓	✓	x	Rastrerizer
4…5	Scissor and Stipple	✓	✓	x	Scissor and Stipple functions
6…7	Texture	✓	✓	x	
8…9	LUT	✓	✓	x	
10…11	Framebuffer	✓	✓	x	
12…13	LogicalOps	✓	✓	x	
14…15	Framebuffer	✓	✓	x	
16…17	Setup	✓	✓	x	
18…30	Reserved	0	0	x	Reserved
31	Global/local toggle	✓	✓	x	selects global (0) or individual settings (1)

Notes: Two bit pixel size encoding: This field sets the pixel size to be used for merging the pixel data into the memory. It is normally set to the same value for all functions, but for generating texture maps it may be advantageous to use a different write pixel size.

- The pixel size is taken from bits 0…1 when bit 31 is 0 or taken from subsequent bites for local functionality when bit 31 is 1.
- The two bit pixel size is encoded as follows:
  0 = 32 bpp 1 = 16 bpp 2 = 8 bpp
- During readback bits 0…17 and 31 return values as loaded and bits 18…30 return zero.

PointTable[0…3]

Name	Type	Offset	Format
PointTable[0…3]	Rasterizer *Control registers*	0x8080, 0x8088, 0x8090, 0x8098	bitfield

Bits	Name	Read	Write	Reset	Description
0…31	PointTable	✓	✓	x	8 delta values 0…7 in fixed point 1.3 format

Notes: Antialiased point data table. There are 4 words in the table of packed dx point data. The format is unsigned 1.3 fixed point numbers. From the host's view the table is organised as 4 * 32 bit words to minimize download overhead when points size changes. Only the parts of the table needed for a particular point size need to be loaded.

ProvokingVertex

Name	Type	Offset	Format
ProvokingVertex	Delta Control register	0x9338	Bitfield

Bits	Name	Read	Write	Reset	Description
0…1	Vertex	✓	✓	x	Data field 0, 1 or 2 for vertex to use for certain parameters
2…31	Reserved	0	0	x	

Notes: If UseProvoking vertex is enabled, certain parameters (defined by the ProvokingVertexMask) are flat shaded using the vertex specified by the provoking vertex register. Flat shaded primitives take the values to be used across the whole primitive from one of the vertices, known as the provoking vertex. Which vertex this is depends on the type of primitive being drawn. The Input unit breaks complex primitives (strips, fans, meshes, etc) into single traingles. It also issues a provoking vertex command which the Delta unit uses as the basis for selecting the vertex from which to take the shading parameters.

ProvokingVertexMask

Name	Type	Offset	Format
ProvokingVertexMask	Delta Control register	0x9358	Bitfield

Bits	Name	Read	Write	Reset	Description
0	R	✓	✓	x	Red
1	G	✓	✓	x	Green
2	B	✓	✓	x	Blue
3	A	✓	✓	x	Alpha
4	Reserved	0	0	x	
5	KsR	✓	✓	x	Red specular component
6	KsG	✓	✓	x	Green specular component
7	KsB	✓	✓	x	Blue specular component
8	Reserved	0	0	x	
9	KdR	✓	✓	x	Red diffuse component
10	KdG	✓	✓	x	Green diffuse component
11	KdB	✓	✓	x	Blue diffuse component
12-31	Reserved	0	0	x	

Notes: If UseProvoking vertex is enabled, certain parameters (defined by the ProvokingVertexMask) are flat shaded using the vertex specified by the provoking vertex register.

The mask is used to select which parameters are constant and should have the deltas set to zero, and which should be interpolated.

Q1Start

Name	Type	Offset	Format
Q1Start	Texture *Control register*	0x8430	Fixed point

Bits	Name	Read	Write	Reset	Description
0…n	Fraction	✓	✓	x	
n…31	Integer	✓	✓	x	

Notes: Initial Q1 value for texture map. The format is 32 bit 2's complement fixed point numbers. The binary point is at an arbitrary location but must be consistent for all S1, T1 and Q1 values.

QStart

Name	**Type**	**Offset**	**Format**
QStart	Texture *Control register*	0x83B8	Fixed point

Bits	**Name**	**Read**	**Write**	**Reset**	**Description**
0…n	Fraction	✓	✓	x	
n…31	Integer	✓	✓	x	

Notes: Initial Q value for texture map. The format is 32 bit 2's complement fixed point numbers. The binary point is at an arbitrary location but must be consistent for all S, T and Q values.

RasterizerMode
RasterizerModeAnd
RasterizerModeOr

Name	Type	Offset	Format
RaasterizerMode	Rasterizer	0x80A0	Bitfield
RaasterizerModeAnd	Rasterizer	0xABA0	Bitfield
RaasterizerModeOr	Rasterizer	0xABA8	Bitfield
	Control register		

Bits	Name	Read [26]	Write	Reset	Description
0	MirrorBit Mask	✓	✓	x	• When set the bit mask bits are consumed from the most significant end towards the least significant end. • When reset the bit mask bits are consumed from the least significant end towards the most significant end.
1	InvertBit Mask	✓	✓	x	When this bit is set the bit mask is inverted first before being tested.
2,3	Fraction Adjust	✓	✓	x	These bits control the action of a ContinueNewLine cmmand and specify how the fraction bits in the Y and XDom DDAs are adjusted. 0: No adjustment is done, 1: Set the fraction bits to zero, 2: Set the fraction bits to half. 3: Set the fraction to *nearly half*, i.e. 0x7fff
4,5	Bias Coordinates	✓	✓	x	These bits control how much is added onto the SartXDom, StartXSub and StartY values when they are loaded into the DDA units. The original registers are not affected. 0: Zero is added, 1: Half is added, 2: *Nearly half*, i.e. 0x7fff is added
6		✓	✓	x	Reserved
7,8	BitMask ByteSwap Mode	✓	✓	x	These bit controls the byte swapping of the BitMask data before it is used. If the bytes are labelled ABCD on input then they are swapped as follows: 0: ABCD (i.e. no swap) 1: BADC 2: CDAB 3: DCBA
9	BitMask Packing	✓	✓	x	This bit controls whether the bitMask data is packed or if a new BitMask data is required on every scanline. 0: BitMask data is packed, 1: BitMask data is provided for each scanline.

[26] Logic Op register readback is via the main register only

10-14	BitMaskOffset	✓	✓	x	These bits hold the bit position in the BitMask data where the first bit is taken from for the bit mask test for the first BitMask data on a new scanline. Subsequent BitMask data starts from bit 0 until the next scanline. Successive bits are taken from increasing bit positions until the bit mask is consumed (i.e. bit 31 is reached). The least significant bit is bit zero.
15,16	HostDataByteSwapMode	✓	✓	x	These bits controls the byte swapping of the BitMask data before it is used. If the bytes are labelled ABCD on input then they are swapped as follows: 0: ABCD (i.e. no swap) 1: BADC 2: CDAB 3: DCBA
17	MultiGLINT	✓	✓	x	This bit selects whether the rasterizer is to work in single GLINT mode, or in multi-GLINT mode and consequently only process the scanlines allocated to it. 0: Single GLINT mode 1: Multi-GLINT mode
18	YLimitsEnable	✓	✓	x	This bit, when set, enables the Y limits testing to be done between the minimum and maximum Y values given by the YLimits register.
19	Reserved	✓	✓	x	
20…22	StripeHeight	✓	✓	x	This field specifies the number of scanlines in a stripe. The options are: 0 = 1 3 = 8 1 = 2 4 = 16 2 = 4
23	WordPacking	✓	✓	x	This bit controls how the two host words sent during , a span operation are packed into the 64 bit internal span data. 0 = first word in bits 0…31, second word in 32…63 1 = first word in bits 32…63, second word in 0…31
24	OpaqueSpans	✓	✓	x	This bit, when set allows the color of each pixel in the span to be either foreground or background as set by the supplied bit masks. If this bit is 0 then any supplied bit masks are anded with the pixel mask to delete pixels from the span. This bit should be set to 0 for performance reasons when foreground/background processing is not required.
25	Reserved	0	0	x	
26	D3DRules	✓	✓	x	This bit, if set, uses D3D rules for subpixel correction calculations, otherwise OpenGL rules are used.
27…31	Reserved	0	0	x	Reserved for future use, mask to 0

Notes: Defines the long term mode of operation of the rasterizer.

The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

RectanglePosition

Name	Type	Offset	Format
RectanglePosition	2DSetup *Control register*	0xB600	Integer

Bits	Name	Read	Write	Reset	Description
0…15	X offset	✓	✓	x	2’s complement X coordinate
16…31	Y offset	✓	✓	x	2’s complement Y coordinate

Notes: This register defines the rectangle origin for use by the Render2D command.

Render

Name	**Type**	**Offset**	**Format**
Render	Global *Command*	0x8038	Bitfield

Bits	Name	Read	Write	Reset	Description
0	AreaStipple Enable	✗	✓	x	This bit, when set, enables area stippling of the fragments produced during rasterisation in the Stipple Unit. Note that area stipple in the Stipple Unit must be enabled as well for stippling to occur. When this bit is reset no area stippling occurs irrespective of the setting of the area stipple enable bit in the Stipple Unit. This bit is useful to temporarily force no area stippling for this primitive.
1	LineStipple Enable	✗	✓	x	This bit, when set, enables line stippling of the fragments produced during rasterisation in the Stipple Unit. Note that line stipple in the Stipple Unit must be enabled as well for stippling to occur. When this bit is reset no line stippling occurs irrespective of the setting of the line stipple enable bit in the Stipple Unit. This bit is useful to temporarily force no line stippling for this primitive.
2	ResetLine Stipple	✗	✓	x	This bit, when set, causes the line stipple counters in the Stipple Unit to be reset to zero, and would typically be used for the first segment in a polyline. This action is also qualified by the LineStippleEnable bit and also the stipple enable bits in the Stipple Unit. When this bit is reset the stipple counters carry on from where they left off (if line stippling is enabled)
3	FastFillEnable	✗	✓	x	This bit, when set, causes the span fill mechanisms to be used for the rasterisation process. The type of span filling is specified in the SpanOperation field. When this bit is reset the normal rasterisation process occurs.
4, 5	Un used	0	0	x	
6, 7	Primitive Type	✗	✓		This two bit field selects the primitive type to rasterise. The primitives are: 0 = Line 1 = Trapezoid 2 = Point
8	Antialiase Enable	✗	✓		This bit, when set, causes the generation of sub scanline data and the coverage value to be calculated for each fragment. The number of sub pixel samples to use is controlled by the AntialiasingQuality bit. When this bit is reset normal rasterisation occurs.
9	Antialiasing Quality	✗	✓		This bit, when set, sets the sub pixel resolution to be 8x8 When this bit is reset the sub pixel resolution is 4x4.

10	UsePoint Table	✗	✓		When this bit and the AntialiasingEnable are set, the dx values used to move from one scanline to the next are derived from the Point Table.
11	SyncOnBit Mask	✗	✓	x	This bit, when set, causes a number of actions: The least significant bit or most significant bit (depending on the MirrorBitMask bit) in the Bit Mask register is extracted and optionally inverted (controlled by the InvertBitMask bit). If this bit is 0 then any fragments are skipped. After every fragment the BitMask register is rotated by one bit. If all the bits in the BitMask register have been used then rasterisation is suspended until a new BitMaskPattern tag is received. If any other tag is received while the rasterisation is suspended then the rasterisation is aborted. The message which caused the abort is then processed as normal. Note the behaviour is slightly different when the SyncOnHostData bit is set to prevent a deadlock from occurring. In this case the rasterisation doesn't suspend when all the bits have been used and if new BitMaskPattern tags are not received in a timely manner then the subsequent fragments will just reuse the bit mask.
12	SyncOnHost Data	✗	✓		When this bit is set a fragment is produced only when one of the following tags have been received from the host: Depth, Stencil, Color or FBData, FBSourceData. If SyncOnBitMask is reset then any tag other than one of these three is received then the rasterisation is aborted. If SyncOnBitMask is set then any tag other than one of these five or BitMaskPattern is received then the rasterisation is aborted. The tag which caused the abort is then processed as normal for that register type**.** The *BitMaskPattern* register doesn't cause any fragments to be generated, but just updates the BitMask register.
13	TextureEnable	✗	✓	x	This bit, when set, enables texturing of the fragments produced during rasterisation. Note that the Texture Units must be suitably enabled as well for any texturing to occur. When this bit is reset no texturing occurs irrespective of the setting of the Texture Unit controls. This bit is useful to temporarily force no texturing for this primitive.
14	FogEnable	✗	✓	x	This bit, when set, enables fogging of the fragments produced during rasterisation. Note that the Fog Unit must be suitably enabled as well for any fogging to occur. When this bit is reset no fogging occurs irrespective of the setting of the Fog Unit controls. This bit is useful to temporarily force no fogging for this primitive.

15	Coverage Enable	✕	✓	x	This bit, when set, enables the coverage value produced as part of the antialiasing to weight the alpha value in the alpha test unit. Note that this unit must be suitably enabled as well. When this bit is reset no coverage application occurs irrespective of the setting of the AntialiasMode.
16	SubPixel Correction Enable	✕	✓	x	This bit, when set enables the sub pixel correction of the color, depth, fog and texture values at the start of a scanline. When this bit is reset no correction is done at the start of a scanline. Sub pixel corrections are only applied to aliased trapezoids.
17	Reserved	0	0	x	
18	SpanOperation	✕	✓	x	This bit, when clear, indicates the writes are to use the constant color found in the previous FBBlockColor register. When this bit is set write data is variable and is either provided by the host (i.e. SyncOnHostData is set) or is read from the framebuffer.
19	Unused	0	0	x	
20…26	Reserved	✕	✓	x	
27	FBSourceRead Enable	✕	✓	x	This bit, when set enables source buffer reads to be done in the Framebuffer Read Unit. Note that the Framebuffer Read Unit must be suitably enabled as well for the source read to occur. When this bit is reset no source reads occur irrespective of the setting of the Framebuffer Read Unit controls.
28…31	Unused	0	0	x	

Notes:

Render2D

Name	Type	Offset	Format
Render2D	Global *Control register*	0xB640	Bitfield

Bits	Name	Read	Write	Reset	Description
0…11	Width	✗	✓	x	Specifies the width of the rectangle in pixels. Its range is 0…4095.
12…13	Operation	✗	✓	x	This two bits field is encoded as follows: 0 = Normal 1 = SyncOnHostData 2 = SyncOnBitMask 3 = PatchOrderRendering The SyncOnHostData and SyncOnBitMask settings just set the corresponding bit in the Render command. PatchOrderRendering decomposes the input rectangle in to a number of smaller rectangels to make better use of the page structure of patched memory.
14	FBRead SourceEnable	✗	✓	x	This bit sets the FBReadSourceEnable bit in the Render command.
15	SpanOperation	✗	✓	x	This bit sets the SpanOperation bit in the Render command.
16…27	Height	✗	✓	x	Specifies the height of the rectangle in pixels. Its range is 0…4095.
28	Increasing X when set	✗	✓	x	This bit, when set, specifies the rasterisation is to be done in increasing X direction.
29	Increasing Y when set	✗	✓	x	This bit, when set, specifies the rasterisation is to be done in increasing Y direction.
30	AreaStipple Enable	✗	✓	x	This bit sets the AreaStippleEnable bit in the Render command.
31	TextureEnable	✗	✓	x	This bit sets the TextureEnable bit in the Render command.

Notes: This command starts a rectangle being rendered from the origin given by the *RectanglePosition* register.

Render2DGlyph

Name	Type	Offset	Format
Render2DGlyph	Global *Command*	0xB648	Bitfield

Bits	Name	Read	Write	Reset	Description
0…6	Width	✗	✓	x	
7…13	Height	✗	✓	x	
14…22	X	✗	✓	x	Signed advance in X
23…31	Y	✗	✓	x	Signed advance in Y

Notes: This command starts a glyph being rendered from the position given by (GlyphPosition+Advance(X, Y)).

RenderPatchOffset

Name	Type	Offset	Format
RenderPatchOffset	Delta *Control register*	0xB610	Bitfield

Bits	Name	Read	Write	Reset	Description
0…15	X coordinate	✓	✓	x	2's complement X coordinate
16…31	Y coordinate	✓	✓	x	2's complement Y coordinate

Notes: This register holds the amount needed to add to the rectangle origin to recover the memory page alignment for the rectangle when it is rendered in patch order.

RepeatLine

Name	Type	Offset	Format
RepeatLine	Delta *Command*	0x9328	Tag

Bits	Name	Read	Write	Reset	Description
0…31	Reserved	0	0	x	

Notes: This command causes the previous line drawn with a DrawLine command to be repeated. It would be normal for some mode or other state information to have been changed before the line is repeated. An example of this is to use scissor clipping with the line being repeated for each clip rectangle.

The data field used when this command is turned into the *Render command* is taken from the previous Draw register.

RepeatTriangle

Name	Type	Offset	Format
RepeatTriangle	Delta *Command*	0x9310	Tag

Bits	Name	Read	Write	Reset	Description
0…31	Reserved	0	0	x	

Notes: This command causes the previous triangle drawn with **DrawTriangle** to be repeated. It would be normal for some mode or other state information to have been changed before the triangle is repeated. An example of this is to use scissor clipping with the triangle being repeated for each clip rectangle. The data field used when this command is turned into the *Render command* is taken from the last Draw register.

ResetPickResult

Name	Type	Offset	Format
ResetPickResult	Output *Command*	0x8C20	Tag

Bits	Name	Read	Write	Reset	Description
0…31	Reserved	0	0	x	

Notes: This register resets the picking result flag. Data field is not used.

RetainedRender

Name	Type	Offset	Format
RetainedRender	Input *Command*	0xB7A0	Bitfield

Bits	Name	Read	Write	Reset	Description
0…31	Command	✗	✓	X	Same as *Render command* format

Notes: See *Render* command.

RLCount

Name	Type	Offset	Format
RLCount	2DSetup *Control register*	0xB678	Integer

Bits	Name	Read	Write	Reset	Description
0…23	Count	✗	✓	x	
24…31	Reserved	0	0	x	

Notes: This register starts the run length expansion being done. The data in RLData is written to the register defined in *DownloadTarget* **count** times. The count is held in bits 0…23 of this command.

RLData

Name	Type	Offset	Format
RLData	Delta *Control register*	0xB670	Integer

Bits	Name	Read	Write	Reset	Description
0…31	RLData	✓	✓	x	32 bit value

Notes: This register holds the 32 bits of data to be repeated when the run length decoding is initiated by the RLCount command.

RLEMask

Name	Type	Offset	Format
RLEMask	Output *Control register*	0x8C48	Bitfield

Bits	Name	Read	Write	Reset	Description
0…31	Mask	✓	✓	0	Mask Data

Notes: This register holds the mask to AND with the run length encoded data and allows bits to be discounted from the comparison. It also sets the unwanted bits to zero in the data value returned with the run length.

RouterMode

Name	Type	Offset	Format
RouterMode	Router *Control register*	0x8840	Bitfield

Bits	Name	Read	Write	Reset	Description
0	Sequence	✓	✓	x	Bit0 may be: 0=Texture, Depth; or 1=Depth, Texture
1…31	Reserved	0	0	x	

Notes: Switches the order of some units in the pipeline.

RStart

Name	Type	Offset	Format
RStart	Color *Control register*	0x8780	Fixed point number

Bits	Name	Read	Write	Reset	Description
0…14	Fraction	✓	✓	x	
15…23	Integer	✓	✓	x	
24…31	Unused	0	0	x	

Notes: Used to set the initial Red value for a vertex when in Gouraud shading mode. The value is 24 bit 2's complement fixed point numbers in 9.15 format.

S1Start

Name	Type	Offset	Format
S1Start	Texture *Control register*	0x8400	Fixed point

Bits	Name	Read	Write	Reset	Description
0…n	Fraction	✓	✓	x	
n…31	Integer	✓	✓	x	

Notes: Initial S1 value for texture map. The format is 32 bit 2's complement fixed point numbers. The binary point is at an arbitrary location but must be consistent for all S1, T1 and Q1 values.

SaveLineStippleCounters

Name	Type	Offset	Format
SaveLineStippleCounters	Stipple *Command*	0x81C0	tag

Bits	Name	Read	Write	Reset	Description
0…31	Reserved	0	0	x	

Notes: Copies the current counter values into an internal register for later restoration using the *UpdateLineStippleCounters* command. Useful in drawing stippled wide lines.

ScissorMaxXY

Name	Type	Offset	Format
ScissorMaxXY	Scissor ***Control register***	0x8190	Bitfield

Bits	Name	Read	Write	Reset	Description
0…15	X coordinate	✓	✓	x	2's complement fixed point X coordinate
16…31	Y coordinate	✓	✓	x	2's complement fixed point Y coordinate

Notes: This register holds the maximum XY scissor coordinate - i.e. the rectangle corner farthest from the screen origin.

ScissorMinXY

Name	Type	Offset	Format
ScissorMinXY	Scissor *Control register*	0x8188	Bitfield

Bits	Name	Read	Write	Reset	Description
0…15	X coordinate	✓	✓	x	2's complement fixed point X coordinate
16…31	Y coordinate	✓	✓	x	2's complement fixed point Y coordinate

Notes: This register holds the minimum XY scissor coordinate - i.e. the rectangle corner closest to the screen origin.

ScissorMode
ScissorModeAnd
ScissorModeOr

Name	Type	Offset	Format
ScissorMode	Scissor	0x8180	Bitfield
ScissorModeAnd	Scissor	0xABB0	Bitfield Logic Mask
ScissorModeOr	Scissor	0xABB8	Bitfield Logic Mask
	Control registers		

Bits	Name	Read [27]	Write	Reset	Description
0	UserScissor Enable	✓	✓	x	enables the user scissor clipping
1	ScreenScissor Enable	✓	✓	x	enables the screen scissor clipping
2…31	Unused	0	0	x	

Notes: Controls enabling of the screen and user scissor tests. The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

ScreenSize

Name	Type	Offset	Format
ScreenSize	Scissor	0x8198	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…15	Width	✓	✓	x	
16…31	Height	✓	✓	x	

Notes: Screen dimensions for screen scissor clipping. The screen boundaries are (0,0) to (width-1, height-1) inclusive.

[27] Logic Op register readback is via the main register only

Security

Name	Type	Offset	Format
Security	Input *Control register*	0x8908	Bitfield

Bits	Name	Read	Write	Reset	Description
0	Secure	✗	✓	x	0 = normal mode 1 = secure mode
1…31	Reserved	0	0	x	

Notes: This unit controls the security of the rest of the pipeline by filtering out any register loads that may cause the pipeline to lockup if used incorrectly. If the security mode is Enable, potentially dangerous registers can only be programmed by a direct write to the register, and not through DMA. This avoids the danger of DMA buffers in user address space being corrupted by another application and causing the chip to lockup. The following registers are filtered out of DMA command buffers if the security bit is enabled:

- FilterMode
- VTGAddress
- VTGData
- Security
- DMARectangleWrite
- DMAOutputCount
- DMAFeedback
- ContextDump
- ContextRestore
- ContextData

SetLogicalTexturePage

Name	Type	Offset	Format
SetLogicalTexturePage	Texture Control *register*	0xB360	Bitfield

Bits	Name	Read	Write	Reset	Description
0…15	PageNumber	✓	✓	x	Logical page number
16…31	Unused	0	0	x	

Notes: This register sets the logical page number to be used in subsequent *UpdateLogicalTextureInfo* commands. The logical page is held in bits 0…15.

SizeOfFramebuffer

Name	Type	Offset	Format
SizeOfFramebuffer	ffer Control register		

Bits	Name	Read	Write	Reset	Description
0…n	Size	✓	✓	x	integer value in units of 16 bytes
n…31	Unused	0	0	x	

Notes: This message holds the size (in units of 16 bytes) of the memory associated with the FB memory interface. Amount of FB Memory SizeOfFramebuffer value:

- 8MBytess shown as 0x80000
- 16MBytes shown as 0x100000

SStart

Name	Type	Offset	Format
SStart	Texture *Control register*	0x8388	Fixed point

Bits	Name	Read	Write	Reset	Description
0…n	Fraction	✓	✓	x	
n…31	Integer	✓	✓	x	

Notes: Initial S value for texture map. The format is 32 bit 2's complement fixed point numbers. The binary point is at an arbitrary location but must be consistent for all S, T and Q values.

StartXDom

Name	Type	Offset	Format
Start X Dominant	Rasterizer *Control register*	0x8000	Fixed point

Bits	Name	Read	Write	Reset	Description
0…15	Fraction	✓	✕	x	
16…31	Integer	✓	✕	x	

Notes: The start X coordinate for the dominant edge: initial X value for the dominant edge in trapezoid filling, or initial X value in line drawing. The value is in 2's complement 16.16 fixed point format..

StartXSub

Name	Type	Offset	Format
Start X Subordinate	Rasterizer *Control register*	0x8010	Fixed point

Bits	Name	Read	Write	Reset	Description
0…15	Fraction	✓	✕	x	
16…31	Integer	✓	✕	x	

Notes: The start X coordinate for the subordinate edge: initial X value for the subordinate edge in trapezoid filling. The value is in 2's complement 16.16 fixed point format.

StartY

Name	**Type**	**Offset**	**Format**
Start Y	Rasterizer *Control register*	0x8020	Fixed point

Bits	Name	Read	Write	Reset	Description
0…15	Fraction	✓	✕	x	
16…31	Integer	✓	✕	x	

Notes: The start Y coordinate: initial scanline (or sub-scanline) in trapezoid filling, or initial Y position for line drawing. The value is in 2's complement 16.16 fixed point format.

StatisticMode
StatisticModeAnd
StatisticModeOr

Name	Type	Offset	Format
StatisticMode	Output	0x8C08	Bitfield
StatisticModeAnd	Output	0xAD10	Bitfield Logic Mask
StatisticModeOr	Output	0xAD18	Bitfield Logic Mask
	Command		

Bits	Name	Read [28]	Write	Reset	Description
0	Enable	✓	✓	x	When set allows the collection of statistics information.
1	StatsType	✓	✓	x	Selects the type of staticstics to gather. The options are: 0 = Picking 1 = Extent
2	ActiveSteps	✓	✓	x	When set includes active fragments in the statistics gathering, otherwise they are excluded.
3	PassiveSteps	✓	✓	x	When set includes culled fragments in the statistics gathering, otherwise they are excluded.
4	Compare Function	✓	✓	x	Selects the type of compare function to use. The options are: 0 = Inside region 1 = Outside region
5	Spans	✓	✓	x	When set includes spans in the statistics gathering, otherwise they are excluded.
6..31	Unused	0	0	x	

Notes: Statistic Collection: here the active fragments and spans are used to (a) record the extent of the rectangular region where rasterization has been occurring, or (b) if rasterization has occurred inside a specific rectangular region. These facilities are useful for picking and debug activities.

Statistic collecting has two modes of operation:

Picking: In this mode the active and/or culled fragments, and spans have the associated XY coordinate compared against the coordinates specified in the *MinRegion* and *MaxRegion* registers. If the result is true then the PickResult flag is set otherwise it holds it previous state. The compare function can be either Inside or Outside. Before picking can start the *ResetPickResult* must be sent to clear the PickResult flag.

Extent: In this mode the active and/or culled fragments and spans have the associated XY coordinates compared to the *MinRegion* and *MaxRegion* registers and if found to be outside the defined rectangular region the appropriate register is updated with the new coordinate(s) to extend the region. The Inside/Outside bit has no effect in this mode.

The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

[28] Logic Op register readback is via the main register only

Stencil

Name	Type	Offset	Format
Stencil	Stencil	0x8998	Bitfield
	Command/control register		

Bits	Name	Read	Write	Reset	Description
0…7	Stencil value	✓	✓	x	8 bit stencil value
8…31	Reserved	0	0	x	

Notes: The ***Stencil*** register holds an externally sourced stencil value. It is a 32 bit register of which only the least significant 8 bits are used. The unused most significant bits should be set to zero. Set the register to the 8 bit stencil value to be used in clearing down the stencil buffer, or in drawing a primitive where the host supplies the stencil value.

StencilData
StencilDataAnd
StencilDataOr

Name	Type	Offset	Format
StencilData	Stencil	0x8990	Bitfield
StencilDataAnd	Stencil	0xB3E0	Bitfield Logic Mask
StencilDataOr	Stencil	0xB3E8	Bitfield Logic Mask
	Control registers		

Bits	Name	Read [29]	Write	Reset	Description
0…7	Stencil value	✓	✓	x	8 bit stencil test value
8…15	Compare mask	✓	✓	x	Determines which bits are significant in the test
16…23	Writemask	✓	✓	x	Determines which bits in localbuffer are updated
24…31	Reserved	0	0	x	

Notes: The register holds data used in the Stencil test:

- Stencil value is the reference value for the stencil test.
- Compare mask is used to determine which bits are significant in the stencil test comparison.
- The stencil writemask is used to control which stencil planes are updated as a result of the test.

The stencil unit must be enabled to update the stencil buffer. If it is disabled then the stencil buffer will only be updated if ForceLBUpdate is set**.** The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

[29] Logic Op register readback is via the main register only

StencilMode
StencilModeAnd
StencilModeOr

Name	Type	Offset	Format
StencilMode	Stencil	0x8988	Bitfield
StencilModeAnd	Stencil	0xAC60	Bitfield Logic Mask
StencilModeOr	Stencil	0xAC68	Bitfield Logic Mask
	Control registers		

Bits	Name	Read [30]	Write	Reset	Description
0	Unitenable	✓	✓	x	0 = Disable 1 = Enable
1…3	Update method	✓	✓	x	if Depth test passes and Stencil test passes (see table 1)
4…6	Update method	✓	✓	x	if Depth test fails and Stencil test passes (see table 1)
7…9	Update method	✓	✓	x	if Stencil test fails (see table 1)
10…12	Mode 0-7	✓	✓	x	Unsigned comparison function (see table 2)
13…14	Stencil source	✓	✓	x	0 = Test Logic 1 = Stencil Register 2 = LBData 3 = LBSourceData
15…16	Stencil widths	✓	✓	x	0 = 4 bits 1 = 8 bits 2 = 1 bit
17…31	Unused	0	0	x	

Notes: Controls the stencil test, which conditionally rejects fragments based on the outcome of a comparison between the value in the stencil buffer and a reference value in the *StencilData* register. If the test is LESS and the result is true then the fragment value is less than the source value..

The logic operator equivalents behave the same way but the new mode is AND’d or OR’d with the former mode before replacing it.

[30] Logic Op register readback is via the main register only

Table 1 – Update Method if Stencil Test fails

Mode	Method	Result
0	Keep	Source stencil
1	Zero	0
2	Replace	Reference stencil
3	Increment	Clamp (Source stencil + 1) to $2^{\text{stencil width}} - 1$
4	Decrement	Clamp (Source stencil -1) to 0
5	Invert	

Table 2 - Unsigned Comparison Function

Mode	Comparison Function
0	NEVER
1	LESS
2	EQUAL
3	LESS OR EQUAL
4	GREATER
5	NOT EQUAL
6	GREATER OR EQUAL
7	ALWAYS

StripeOffsetY

Name	Type	Offset	Format
StripeOffsetY	*Control register*	0x80C8	Fixed point

Bits	Name	Read	Write	Reset	Description
0…15	Fixed point	✓	✓	x	2’s complement fixed point value
16…23	Reserved	0	0	x	Reserved for future use, mask to 0

Notes: This register holds the 16 bit 2's complement Y value added to the raster Y value to determine scanline ownership.

SuspendUntilFrameBlank

Name	Type	Offset	Format
SuspendUntilFrameBlank	Framebuffer *Command*	0x8C78	Bitfield

Bits	Name	Read	Write	Reset	Description
0…20	ScreenBase	✓	✓	x	Base address of screen in 128 bit units
21…31	Reserved	0	0	x	

Notes: The *SuspendUntilFrameBlank* command flushes the write combine buffers and then is forwarded onto the Memory Controller where it prevents any further memory writes (normal or span writes) from this port until after the next the Vertical Frame Blank has happened. When frame blank occurs new writes are allowed to proceed.

By using this register the host does not need to get involved with waiting for vertical frame blank itself before it can issue new instructions to P3. While waiting for frame blank any data or actions which do not involve writing to the memory via this unit (such as clearing down the depth buffer) can proceed. Attempting to write to the memory while waiting for frame blank will just result in the Write FIFO blocking for the duration and this will ripple back through the chip

Sync

Name	Type	Offset	Format
Synchronization	Output *Control register*	0x8C40	Bitfield

Bits	Name	Read	Write	Reset	Description
0…30	User defined	✗	✓	x	User defined
31	Interrupt enable	✗	✓	x	Interrupt after output FIFO write operations

Notes: This command can be used to synchronize with the host. It is also used to flush outstanding operations such as pending memory accesses. It also causes the current status of the picking result to be passed to the Host Out FIFO unless culled by the statistics bits in the *FilterMode* register.

If bit 31 of the input data is set then an interrupt is generated. The data output is the value written to the register by this command. If interrupts are enabled, then the interrupt does not occur until the tag and/or data have been written to the output FIFO.

T1Start

Name	Type	Offset	Format
T1Start	Texture *Control register*	0x8418	Fixed point

Bits	Name	Read	Write	Reset	Description
0…n	Fraction	✓	✓	x	
n…31	Integer	✓	✓	x	

Notes: Initial T1 value for texture map. The format is 32 bit 2's complement fixed point numbers. The binary point is at an arbitrary location but must be consistent for all S1, T1 and Q1 values.

TailPhysicalPageAllocation[0…3]

Name	Type	Offset	Format
TailPhysicalPageAllocation [0…3]	Texture *Control register*	0xB4A0, 0xB4A8, 0xB4B0, 0xB4B8	Integer

Bits	Name	Read	Write	Reset	Description
0…15	Address	✓	✓	x	16 bit value 0…65535

Notes: These registers hold the tail page for memory pools 0…3. This is usually the least recently referenced physical page in the pool of the working set. The range of physical pages is 0…65535.

TextRender2DGlyph0…7

Name	Type	Offset	Format
TextRender2DGlyph0	Global	0x8708	Bitfield
TextRender2DGlyph1	Global	0x8718	Bitfield
TextRender2DGlyph2	Global	0x8728	Bitfield
TextRender2DGlyph3	Global	0x8738	Bitfield
TextRender2DGlyph4	Global	0x8748	Bitfield
TextRender2DGlyph5	Global	0x8758	Bitfield
TextRender2DGlyph6	Global	0x8768	Bitfield
TextRender2DGlyph7	Global	0x8778	Bitfield
	Command		

Bits	Name	Read	Write	Reset	Description
0…6	Width	✕	✓	x	
7…13	Height	✕	✓	x	
14…22	X	✕	✓	x	Signed advance in X
23…31	Y	✕	✓	x	Signed advance in Y

Notes: Alias for Render2Dglyph. This command starts a glyph being rendered from the position given by (GlyphPosition+Advance(X, Y)).

TextGlyphAddr0…7

Name	Type	Offset	Format
TextGlyphAddr0	Texture	0x8700	Integer
TextGlyphAddr1	Texture	0x8710	Integer
TextGlyphAddr2	Texture	0x8720	Integer
TextGlyphAddr3	Texture	0x8730	Integer
TextGlyphAddr4	Texture	0x8740	Integer
TextGlyphAddr5	Texture	0x8750	Integer
TextGlyphAddr6	Texture	0x8760	Integer
TextGlyphAddr7	Texture	0x8770	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…31	Base address	✓	✓	x	32 bit value

Notes: Alias for *TextureBaseAddr0.* These registers hold the base address of each texture map (or level for a mip map). The address should be aligned to the natural size of the texture map, however some layouts impose additional restrictions.

TextureApplicationMode
TextureApplicationModeAnd
TextureApplicationModeOr

Name	Type	Offset	Format
TextureApplicationMode	Texture Application	0x8680	Bitfield
TextureApplication ModeAnd	Texture Application	0xAC50	Bitfield Logic Mask
TextureApplicationModeOr	Texture Application	0xAC58	Bitfield Logic Mask
	Control registers		

Bits	Name	Read [31]	Write	Reset	Description
0	Enable	✓	✓	x	When set causes the output to be calculated as defined by the fields in this register, otherwise the fragment's data is passed through.
1…2	ColorA	✓	✓	x	This field selects the source value for A. The options are: 0 = Color.C 1 = Color.A 2 = K.C (TextureEnvColor) 3 = K.A (TextureEnvColor)
3…4	ColorB	✓	✓	x	This field selects the source value for B. The options are: 0 = Texel.C 1 = Texel.A 2 = K.C (TextureEnvColor) 3 = K.A (TextureEnvColor)
5…6	ColorI	✓	✓	x	This field selects the source value for I. The options are: 0 = Color.A 1 = K.A (TextureEnvColor) 2 = Texel.C 3 = Texel.A
7	ColorInvertI	✓	✓	x	This bit, if set, will invert the selected I value before it is used.

[31] Logic Op register readback is via the main register only

8…10	Color Operation	✓	✓	x	This field defines how the three inputs (A, B and I) are combined. Note the I inputs can be optionally inverted before being combined. The 8 bit inputs are unsigned 0.8 fixed point format, but 255 is treated as if it were 1.0 for the calculations. The possible operations are: 0 = PassA (A) 1 = PassB (B) 2 = Add (A + B) 3 = Modulate (A * B) 4 = Lerp (A * (1.0 - I) + B * I) 5 = ModulateColorAddAlpha (A * B + I) 6 = ModulateAlphaAddColor (A * I + B) 7 = ModulateBIAddA (B * I + A)
11…12	AlphaA	✓	✓	x	This field selects the source value for A. The options are: 0 = Color.C (effectively Color.A) 1 = Color.A 2 = K.C (TextureEnvColor) (effectively K.A) 3 = K.A (TextureEnvColor)
13…14	AlphaB	✓	✓	x	This field selects the source value for B. The options are: 0 = Texel.C (effectively T.A) 1 = Texel.A 2 = K.C (TextureEnvColor) (effectively K.A) 3 = K.A (TextureEnvColor)
15…16	AlphaI	✓	✓	x	This field selects the source value for I. The options are: 0 = Color.A 1 = K.A (TextureEnvColor) 2 = Texel.C (effectively T.A) 3 = Texel.A
17	Alpha InvertI	✓	✓	x	This bit, if set, will invert the selected I value before it is used.
18…20	Alpha Operation	✓	✓	x	This field defines how the three inputs (A, B and I) are combined. Note the I inputs can be optionally inverted before being combined. The 8 bit inputs are unsigned 0.8 fixed point format, but 255 is treated as if it were 1.0 for the calculations. The possible operations are: 0 = PassA (A) 1 = PassB (B) 2 = Add (A + B) 3 = Modulate (A * B) 4 = Lerp (A * (1.0 - I) + B * I) 5 = ModulateABAddI (A * B + I) 6 = ModulateAIAddB (A * I + B) 7 = ModulateBIAddA (B * I + A)
21	KdEnable	✓	✓	x	When set this bit causes the RGB results of the texture application to be multiplied by the Kd DDA values. It also enables the Kd DDA sto be updated.

22	KsEnable	✓	✓	x	When set this bit causes the RGB results of the texture application (or Kd processing) to be added wi th the Ks DDA values. It also enables the Ks DDAs to be updated.
23	Motion Comp Enable	✓	✓	x	This bit, when set causes the color field to be interpreted as holding YUV difference values as three 9 bit 2's complement numbers. These are subtracted from the RGB channels of the texel value (after all previous processing) and the result clamped. This is used as part of MPEG Motion Compensation processing.
24…31	Unused	0	0	x	

Notes: Formerly known as *TextureColorMode*. Defines the operation for the color channels in applying texture. Note that the TextureEnable bit in the *Render* command must be set for a primitive to be texture mapped.

The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

TextureBaseAddr[0…15]

Name	**Type**	**Offset**	**Format**
Texture Base Address [0…15]	Texture	0x8500	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…31	Address	✓	✓	x	32 bit value

Notes: This register holds the base address of each texture map (or level for a mip map). The address should be aligned to the natural size of the texture map, however some layouts impose additional restrictions.

The MapBaseRegister field of the *TextureReadMode* register defines which TextureBaseAddr register should be used to hold the address for map level 0 when mip mapping, or the texture map when not mip mapping. Successive map levels are at increasing *TextureBaseAddr* registers upto (and including) the MapMaxLevel. 3D textures always use *TextureBaseAddr0*.

TextureCacheReplacementMode

Name	Type	Offset	Format
TextureCacheReplacement Mode	Input	0xB430	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	KeepOldest0	✓	✓	X	This bit, when set, will keep the oldest texels on the scanline when the cache bank 0 is about to wrap and just re-use a set of scratch lines.
1…5	ScratchLines0	✓	✓	X	This field holds the number of cache lines to use as scratch lines when the cache bank 0 wraps and the KeepOldest mode bit is set. The value in this field has a MIN_SCRATCH_SIZE value (currently 8) added to it so we can guarantee the scratch line size can always accommodate the cache lines the current fragments requires with some left over. Failure to make this provision would lead to deadlock.
6	KeepOldest1	✓	✓	X	This bit, when set, will keep the oldest texels on the scanline when the cache bank 1 is about to wrap and just re-use a set of scratch lines.
7…11	ScratchLines1	✓	✓	X	This field holds the number of cache lines to use as scratch lines when the cache bank 1 wraps and the KeepOldest mode bit is set. The value in this field has a MIN_SCRATCH_SIZE value (currently 8) added to it so we can guarantee the scratch line size can always accommodate the cache lines the current fragments requires with some left over. Failure to make this provision would lead to deadlock.

Notes: This command defines the replacement mode for the two banks of the cache.

TextureChromaLower0
TextureChromaUpper0

Name	Type	Offset	Format
TextureChromaLower0	Texture	0x84F0	Bitfield
TextureChromaUpper0		0x84E8	
	Control register		

Bits	Name	Read	Write	Reset	Description
0..7	R	✓	✓	x	Red
8..15	G	✓	✓	x	Green
16..23	B	✓	✓	x	Blue

24..31	A	✓	✓	x	Alpha

Notes: These registers hold the lower and upper chroma colors to use when the chroma test is enabled for texels from texture map 0. The format is 8 bit ABGR components packed into a 32 bit word with R in the ls byte.

TextureChromaUpper1
TextureChromaLower1

Name	Type	Offset	Format
TextureChromaUpper1	Texture	0x8600	Bitfield
TextureChromaLower1	Texture	0x8608	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0..7	R	✓	✓	x	Red
8..15	G	✓	✓	x	Green
16..23	B	✓	✓	x	Blue
24..31	A	✓	✓	x	Alpha

Notes: These registers hold the upper and lower chroma colors to use when the chroma test is enabled for texels T4…T7. Its format is 8 bit ABGR components packed into a 32 bit word with R in the ls byte.

TextureCompositeAlphaMode0
TextureCompositeAlphaMode0And
TextureCompositeAlphaMode0Or

Name	Type	Offset	Format
TextureCompositeAlphaMode0	Texture	0xB310	Bitfield
TextureCompositeAlphaMode0And	Texture	0xB390	Bitfield Logic Mask
TextureCompositeAlphaMode0Or	Texture	0xB398	Bitfield Logic Mask
	Control registers		

Bits	Name	Read	Write	Reset	Description
0	Enable	✓	✓	ǀ	When set causes the output to be calculated as defined by the fields in this register, otherwise the texel0 data is passed through for stage0 and Output data is passed through for stage 1.
1…4	Arg1	✓	✓	ǀ	This field selects the source value for Arg1. The options are: 0 = Output.C of the previous stage or height if the first stage 1 = Output.A of the previous stage or height if the first stage 2 = Color.C 3 = Color.A 4 = TextureCompositeFactor0.C 5 = TextureCompositeFactor0.A 6 = Texel0.C 7 = Texel0.A 8 = Texel1.C 9 = Texel1.A 10 = Sum of the color components of the previous stage or 0 if the first stage. where C is the RGB or A depending on the channel. height is defined as clamp (Texel0.A - Texel1.A + 128)
5	InvertArg1	✓	✓	x	This bit, if set, will invert the selected Arg1 value before it is used.

6…9	Arg2	✓	✓	x	This field selects the source value for Arg2. The options are: 0 = Output.C of the previous stage or height if the first stage 1 = Output.A of the previous stage or height if the first stage 2 = Color.C 3 = Color.A 4 = TextureCompositeFactor0 C 5 = TextureCompositeFactor0 A 6 = Texel0.C 7 = Texel0.A 8 = Texel1.C 9 = Texel1.A 10 = Sum of the color components of the previous stage or 0 if the first stage. …where C is the RGB or A depending on the channel, and height is defined as clamp (Texel0.A - Texel1.A + 128)
10	InvertArg2	✓	✓	x	This bit, if set, will invert the selected Arg2 value before it is used.
11…13	I	✓	✓	x	This field selects what is used as the interpolation factor when the Operation field is set to Lerp, for example. The options are: 0 = Output.A of the previous stage or 0 if the first stage 1 = Color.A 2 = TextureCompositeFactor0.A 3 = Texel0.A 4 = Texel1.A where C is the RGB or A depending on the channel.
14	InvertI	✓	✓	x	This bit, if set, will invert the selected I value before it is used.
15	A	✓	✓	x	This bit selects which Arg (after any inversion) is to be used as A in the Operation. The options are: 0 = Arg1 1 = Arg2
16	B	✓	✓	x	This bit selects which Arg (after any inversion) is to be used as B in the Operation. The options are: 0 = Arg1 1 = Arg2

17…20	Operation	✓	✓	x	This field defines how the three inputs (A, B and I) are combined. Note the inputs can be optionally inverted before being combined. The 8 bit inputs are unsigned 0.8 fixed point format, but 255 is treated as if it were 1.0 for the calculations. The possible operations are: 0 = Pass (A) 1 = Add (A + B) 2 = AddSigned (A + B - 128) 3 = Subtract (A - B) 4 = Modulate (A * B) 5 = Lerp (A * (1.0 - I) + B * I) 6 = ModulateColorAddAlpha (A * B + I) 7 = ModulateAlphaAddColor (A * I + B) 8 = AddSmoothSaturate (A + B - A * B) 9 = ModulateSigned (A * B, but A and B are biased 8 bit numbers)
21…22	Scale	0	0	x	This field selects the scale factor to apply to the final result before it is clamped. The options are: 0 = 0.5 1 = 1 2 = 2 3 = 4
23…31	Reserved	0	0	x	

Notes: The Texture unit composites the Color, Texel0 and Texel1 fragment's values with one or two constant color values held in registers and passes the result on to the next unit as a texture value.
The compositing is done in two stages and is controlled separately for the RGB channels and the Alpha channel. This register defines the operation for the alpha channels in compositing stage 0 for this unit.

The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

TextureCompositeAlphaMode1
TextureCompositeAlphaMode1And
TextureCompositeAlphaMode1Or

Name	Type	Offset	Format
TextureCompositeAlphaMode1	Texture	0xB320	Bitfield
TextureCompositeAlphaMode1And	Texture	0xB3B0	Bitfield Logic Mask
TextureCompositeAlphaMode1Or	Texture	0xB3B8	Bitfield Logic Mask
	Control registers		

Bits	Name	Read [32]	Write	Reset	Description
0	Enable	✓	✓	x	When set causes the output to be calculated as defined by the fields in this register, otherwise the texel0 data is passed through for stage0 and Output data is passed through for stage 1.
1…4	Arg1	✓	✓	x	This field selects the source value for Arg1. The options are: 0 = Output.C of the previous stage or height if the first stage 1 = Output.A of the previous stage or height if the first stage 2 = Color.C 3 = Color.A 4 = TextureCompositeFactor1C 5 = TextureCompositeFactor1A 6 = Texel0.C 7 = Texel0.A 8 = Texel1.C 9 = Texel1.A 10 = Sum of the color components of the previous stage or 0 if the first stage. where C is the RGB or A depending on the channel. height is defined as clamp (Texel0.A - Texel1.A + 128)
5	InvertArg1	✓	✓	x	This bit, if set, will invert the selected Arg1 value before it is used.

[32] Logic Op register readback is via the main register only

6…9	Arg2	✓	✓	x	This field selects the source value for Arg2. The options are: 0 = Output.C of the previous stage or height if the first stage 1 = Output.A of the previous stage or height if the first stage 2 = Color.C 3 = Color.A 4 = TextureCompositeFactor1C 5 = TextureCompositeFactor1A 6 = Texel0.C 7 = Texel0.A 8 = Texel1.C 9 = Texel1.A 10 = Sum of the color components of the previous stage or 0 if the first stage. where C is the RGB or A depending on the channel. height is defined as clamp (Texel0.A - Texel1.A + 128)
10	InvertArg2	✓	✓	x	This bit, if set, will invert the selected Arg2 value before it is used.
11…13	I	✓	✓	x	This field selects what is used as the interpolation factor when the Operation field is set to Lerp, for example. The options are: 0 = Output.A of the previous stage or 0 if the first stage 1 = Color.A 2 = TextureCompositeFactor1.A 3 = Texel0.A 4 = Texel1.A where C is the RGB or A depending on the channel.
14	InvertI	✓	✓	x	This bit, if set, will invert the selected I value before it is used.
15	A	✓	✓	x	This bit selects which Arg (after any inversion) is to be used as A in the Operation. The options are: 0 = Arg1 1 = Arg2
16	B	✓	✓	x	This bit selects which Arg (after any inversion) is to be used as B in the Operation. The options are: 0 = Arg1 1 = Arg2

17…20	Operation	✓	✓	x	This field defines how the three inputs (A, B and I) are combined. Note the inputs can be optionally inverted before being combined. The 8 bit inputs are unsigned 0.8 fixed point format, but 255 is treated as if it were 1.0 for the calculations. The possible operations are: 0 = Pass (A) 1 = Add (A + B) 2 = AddSigned (A + B - 128) 3 = Subtract (A - B) 4 = Modulate (A * B) 5 = Lerp (A * (1.0 - I) + B * I) 6 = ModulateColorAddAlpha (A * B + I) 7 = ModulateAlphaAddColor (A * I + B) 8 = AddSmoothSaturate (A + B - A * B) 9 = ModulateSigned (A * B, but A and B are biased 8 bit numbers)
21…22	Scale	✓	✓	x	This field selects the scale factor to apply to the final result before it is clamped. The options are: 0 = 0.5 1 = 1 2 = 2 3 = 4
23…31	Reserved	0	0	x	

Notes: The Texture unit composites the fragment's Color, Texel0 and Texel1 values with one or two constant color values held in registers and passes the result on to the next unit as a texture value.
The compositing is done in two stages and is controlled separately for the RGB channels and the Alpha channel. This register defines the operation for the alpha channels in compositing stage 0 for this unit.

The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

TextureCompositeColorMode0
TextureCompositeColorMode0And
TextureCompositeColorMode0Or

Name	Type	Offset	Format
TextureCompositeColorMode0	Texture	0xB308	Bitfield
TextureCompositeColorMode0And	Texture	0xB380	Bitfield Logic Mask
TextureCompositeColorMode0Or	Texture	0xB388	Bitfield Logic Mask
	Control registers		

Bits	Name	Read	Write	Reset	Description
0	Enable	✓	✓	x	When set causes the output to be calculated as defined by the fields in this register, otherwise the texel0 data is passed through for stage0 and Output data is passed through for stage 1.
1…4	Arg1	✓	✓	x	This field selects the source value for Arg1. The options are: 0 = Output.C of the previous stage or height if the first stage 1 = Output.A of the previous stage or height if the first stage 2 = Color.C 3 = Color.A 4 = TextureCompositeFactor0.C 5 = TextureCompositeFactor0.A 6 = Texel0.C 7 = Texel0.A 8 = Texel1.C 9 = Texel1.A 10 = Sum of the color components of the previous stage or 0 if the first stage. where C is the RGB or A depending on the channel. Height is defined as clamp (Texel0.A - Texel1.A + 128)
5	InvertArg1	✓	✓	x	This bit, if set, will invert the selected Arg1 value before it is used.

6…9	Arg2	✓	✓	x	This field selects the source value for Arg2. The options are: 0 = Output.C of the previous stage or height if the first stage 1 = Output.A of the previous stage or height if the first stage 2 = Color.C 3 = Color.A 4 = TextureCompositeFactor0.C 5 = TextureCompositeFactor0.A 6 = Texel0.C 7 = Texel0.A 8 = Texel1.C 9 = Texel1.A 10 = Sum of the color components of the previous stage or 0 if the first stage. where C is the RGB or A depending on the channel. height is defined as clamp (Texel0.A - Texel1.A + 128)
10	InvertArg2	✓	✓	x	This bit, if set, will invert the selected Arg2 value before it is used.
11…13	I	✓	✓	x	This field selects what is used as the interpolation factor when the Operation field is set to Lerp, for example. The options are: 0 = Output.A of the previous stage or 0 if the first stage 1 = Color.A 2 = TextureCompositeFactor*0*.A 3 = Texel0.A 4 = Texel1.A where C is the RGB or A depending on the channel.
14	InvertI	✓	✓	x	This bit, if set, will invert the selected I value before it is used.
15	A	✓	✓	x	This bit selects which Arg (after any inversion) is to be used as A in the Operation. The options are: 0 = Arg1 1 = Arg2
16	B	✓	✓	x	This bit selects which Arg (after any inversion) is to be used as B in the Operation. The options are: 0 = Arg1 1 = Arg2

17…20	Operation	✓	✓	x	This field defines how the three inputs (A, B and I) are combined. Note the inputs can be optionally inverted before being combined. The 8 bit inputs are unsigned 0.8 fixed point format, but 255 is treated as if it were 1.0 for the calculations. The possible operations are: 0 = Pass (A) 1 = Add (A + B) 2 = AddSigned (A + B - 128) 3 = Subtract (A - B) 4 = Modulate (A * B) 5 = Lerp (A * (1.0 - I) + B * I) 6 = ModulateColorAddAlpha (A * B + I) 7 = ModulateAlphaAddColor (A * I + B) 8 = AddSmoothSaturate (A + B - A * B) 9 = ModulateSigned (A * B, but A and B are biased 8 bit numbers)
21…22	Scale	✓	✓	x	This field selects the scale factor to apply to the final result before it is clamped. The options are: 0 = 0.5 1 = 1 2 = 2 3 = 4
23…31	Reserved	0	0	x	

Notes: The Texture unit composites the framgent's Color, Texel0 and Texel1 values with one or two constant color values held in registers and passes the result on to the next unit as a texture value.
The compositing is done in two stages and is controlled separately for the RGB channels and the Alpha channel. This register defines the operation for the alpha channels in compositing stage 0 for this unit.

The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

TextureCompositeColorMode1
TextureCompositeColorMode1And
TextureCompositeColorMode1Or

Name	Type	Offset	Format
TextureCompositeColor Mode1	Texture	0xB318	Bitfield
TextureCompositeColor Mode1And	Texture	0xB3A0	Bitfield Logic Mask
TextureCompositeColor Mode1Or	Texture	0xB3A8	Bitfield Logic Mask
	Control registers		

Bits	Name	Read	Write	Reset	Description
0	Enable	✓	✓	x	When set causes the output to be calculated as defined by the fields in this register, otherwise the texel0 data is passed through for stage0 and Output data is passed through for stage 1.
1…4	Arg1	✓	✓	x	This field selects the source value for Arg1. The options are: 0 = Output.C of the previous stage or height if the first stage 1 = Output.A of the previous stage or height if the first stage 2 = Color.C 3 = Color.A 4 = TextureCompositeFactor1.C 5 = TextureCompositeFactor1.A 6 = Texel0.C 7 = Texel0.A 8 = Texel1.C 9 = Texel1.A 10 = Sum of the color components of the previous stage or 0 if the first stage. where *n* is the same as the message suffix and C is the RGB or A depending on the channel. height is defined as clamp (Texel0.A - Texel1.A + 128)
5	InvertArg1	✓	✓	x	This bit, if set, will invert the selected Arg1 value before it is used.

6…9	Arg2	✓	✓	x	This field selects the source value for Arg2. The options are: 0 = Output.C of the previous stage or height if the first stage 1 = Output.A of the previous stage or height if the first stage 2 = Color.C 3 = Color.A 4 = TextureCompositeFactor1.C 5 = TextureCompositeFactor1.A 6 = Texel0.C 7 = Texel0.A 8 = Texel1.C 9 = Texel1.A 10 = Sum of the color components of the previous stage or 0 if the first stage. where C is the RGB or A depending on the channel. height is defined as clamp (Texel0.A - Texel1.A + 128)
10	InvertArg2	✓	✓	x	This bit, if set, will invert the selected Arg2 value before it is used.
11…13	I	✓	✓	x	This field selects what is used as the interpolation factor when the Operation field is set to Lerp, for example. The options are: 0 = Output.A of the previous stage or 0 if the first stage 1 = Color.A 2 = TextureCompositeFactor1.A 3 = Texel0.A 4 = Texel1.A where C is the RGB or A depending on the channel.
14	InvertI	✓	✓	x	This bit, if set, will invert the selected I value before it is used.
15	A	✓	✓	x	This bit selects which Arg (after any inversion) is to be used as A in the Operation. The options are: 0 = Arg1 1 = Arg2
16	B	✓	✓	x	This bit selects which Arg (after any inversion) is to be used as B in the Operation. The options are: 0 = Arg1 1 = Arg2

17…20	Operation	✓	✓	x	This field defines how the three inputs (A, B and I) are combined. Note the inputs can be optionally inverted before being combined. The 8 bit inputs are unsigned 0.8 fixed point format, but 255 is treated as if it were 1.0 for the calculations. The possible operations are: 0 = Pass (A) 1 = Add (A + B) 2 = AddSigned (A + B - 128) 3 = Subtract (A - B) 4 = Modulate (A * B) 5 = Lerp (A * (1.0 - I) + B * I) 6 = ModulateColorAddAlpha (A * B + I) 7 = ModulateAlphaAddColor (A * I + B) 8 = AddSmoothSaturate (A + B - A * B) 9 = ModulateSigned (A * B, but A and B are biased 8 bit numbers)
21…22	Scale	✓	✓	x	This field selects the scale factor to apply to the final result before it is clamped. The options are: 0 = 0.5 1 = 1 2 = 2 3 = 4
23…31	Reserved	0	0	x	

Notes: The Texture unit composites the fragment's Color, Texel0 and Texel1 values with one or two constant color values held in registers and passes the result on to the next unit as a texture value.
The compositing is done in two stages and is controlled separately for the RGB channels and the Alpha channel. This register defines the operation for the alpha channels in compositing stage 0 for this unit.

The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

TextureCompositeFactor0

Name	Type	Offset	Format
TextureCompositeFactor0	Global *Command*	0xB328	Bitfield

Bits	Name	Read	Write	Reset	Description
0…7	red	✓	✓	x	red
8…15	green	✓	✓	x	green
16…23	blue	✓	✓	x	blue
24…31	alpha	✓	✓	x	alpha

Notes: The Texture unit composites the fragment's Color, Texel0 and Texel1 values with one or two constant color values held in registers and passes the result on to the next unit as a texture value.
The compositing is done in two stages and is controlled separately for the RGB channels and the Alpha channel. This register holds the constant factor to use with compositing stage 0.

TextureCompositeFactor1

Name	Type	Offset	Format
TextureCompositeFactor1	Texture *Command*	0xB330	Bitfield

Bits	Name	Read	Write	Reset	Description
0…7	red	✓	✓	x	red
8…15	green	✓	✓	x	green
16…23	blue	✓	✓	x	blue
24…31	alpha	✓	✓	x	alpha

Notes: The Texture unit composites the Color, Texel0 and Texel1 from a step message with one or two constant color values held in registers and passes the result on to the next unit as a texture value.
The compositing is done in two stages and is controlled separately for the RGB channels and the Alpha channel. This register holds the constant factor to use with compositing stage 1.

TextureCompositeMode

Name	Type	Offset	Format
TextureCompositeMode	Texture *Command*	0xB300	Bitfield

Bits	Name	Read	Write	Reset	Description
0	Enable	✓	✓	x	Global enable/disable for Texture Composition
1…31	Unused	0	0	x	

Notes: Global enable/disable for Texture Composite operation. Setting Bit0 causes the compositing operation to be calculated and to replace the texture0 value sent to the next unit, otherwise the texture value remains unchanged. This enable is also qualified by the TextureEnable bit in the *Render* command.

TextureCoordMode
TextureCoordModeAnd
TextureCoordModeOr

Name	Type	Offset	Format
TextureCoordMode	Texture	0x8380	Bitfield
TextureCoordModeAnd	Texture	0xAC20	Bitfield
TextureCoordModeOr	Texture	0xAC28	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	Enable	✓	✓	x	When set causes the output to be calculated as defined by the fields in this register, otherwise the output values are set to zero. The TextureEnable bit in the Render command must also be set to enable this unit.
1…2	WrapS	✓	✓	x	This field determines how the s coordinate is brought into the range 0.0…1.0 when it is outside this range. The options are: 0 = Clamp 1 = Repeat 2 = Mirror
3…4	WrapT	✓	✓	x	This field determines how the t coordinate is brought into the range 0.0…1.0 when it is outside this range. The options are: 0 = Clamp 1 = Repeat 2 = Mirror
5	Operation	✓	✓	x	This bit selects if the texture coordinates are to be treated as 2D coordinates and ignore perspective correction, or a 3D coordinates and be perspectively corrected. 0 = 2D mode 1 = 3D mode When reset the addresses are calculated in '2D mode' so no perspective correction is done. This will typically run twice as fast as '3D mode' where perspective correction is done. In the 2D case the wrap operation is always “repeat” as the DDA units are allowed to wrap around and have the fixed 0.32 fixed point format. Level of detail calculation is not done in 2D mode.
6	InhibitDDAInitialisation	✓	✓	x	This bit, when set, prevents the DDA from being updated from the Start registers at the start of a primitive. This is useful when the texture mapping is being used to provide the pattern or stipple along a polyline and it is desirable that the pattern continues smoothly from one line to the next.

7	EnableLOD	✓	✓	x	This bit, when set, causes the level of detail calculation to be calculated. This also involves setting the start values of the S1, T1 and Q1 DDAs as a function of the DY gradients and the S, T and Q start values.
8	EnableDY	✓	✓	x	This bit, when set, causes the DY gradients of S, T and Q to be calculated, otherwise they are provided by some external source.
9…12	Width	✓	✓	x	This field holds the width, as a power of 2, of the highest resolution texture map when mip mapping. Its legal range is 0…11 inclusive and is only used when the EnableLOD bit is 1.
13…16	Height	✓	✓	x	This field holds the height, as a power of 2, of the highest resolution texture map when mip mapping. Its legal range is 0…11 inclusive and is only used when the EnableLOD bit is 1.
17	Type	✓	✓	x	This bit selects type of texture map and is only used to disable the t derivatives from influencing the level of detail calculations when a 1D texture map is being used. 0 = 1D map 1 = 2D map
18…19	WrapS1	✓	✓	x	This field determines how the s1 coordinate is brought into the range 0.0…1.0 when it is outside this range. The options are: 0 = Clamp 1 = Repeat 2 = Mirror
20…21	WrapT1	✓	✓	x	This field determines how the t1 coordinate is brought into the range 0.0…1.0 when it is outside this range. The options are: 0 = Clamp 1 = Repeat 2 = Mirror
22	Duplicate Coords	✓	✓	x	This bit, when set, causes any loading one of the DDA start, dx or dyDom registers to load the corresponding registers for both texture 0 and texture 1 DDA
23…31	Uused	0	0	x	

Notes: Provides overall control of the generation of texel addresses.

TextureEnvColor

Name	**Type**	**Offset**	**Format**
TextureEnvironmentColor	Texture *Control register*	0x8688	Bitfield

Bits	Name	Read	Write	Reset	Description
0…7	R	✓	✓	x	Red
8…15	G	✓	✓	x	Green
16…23	B	✓	✓	x	Blue

24…31	A	✓	✓	x	Alpha

Notes: Constant color value used in blend texturing mode..

TextureFilterMode
TextureFilterModeAnd
TextureFilterModeOr

Name	Type	Offset	Format
TextureFilterMode	Alpha Blend	0x84E0	Bitfield
TextureFilterModeAnd	Alpha Blend	0xAD50	Bitfield Logic Mask
ChromaTestModeOr	Alpha Blend	0xAD58	Bitfield Logic Mask
	Control registers		

Bits	Name	Read [33]	Write	Reset	Description
0	Enable	✓	✓	x	When set causes the output to be calculated as defined by the fields in this register, otherwise the texel0 and texel1 values are set to zero. The TextureEnable bit in the *Render* command must also be set to enable this unit.
1…4	Format0	✓	✓	x	This field selects the format of the texel data T0…T3. The options are 0 = A4L4 1 = L8 2 = I8 3 = A8 4 = 332 5 = A8I8 6 = 5551 7 = 565 8 = 4444 9 = 888 10 = 8888 or YUV
5	ColorOrder0	✓	✓	x	This bit selects the color component order of the texel data T0…T3. The two options are: 0 = AGBR 1 = ARGB
6	AlphaMapEnable0	✓	✓	x	This bit, when set, enables the alpha value of texels T0…T3 to be forced to zero based on testing the color values.
7	AlphaMapSense0	✓	✓	x	This bit selects if the alpha value for texels T0…T3 should be set to zero when the colors are in range or out of range. The options are: 0 = Out of range 1 = In range
8	Combine Caches	✓	✓	x	This bit, when set, combines both banks of the cache so they are used for texture 0. This is an optimisation and allows larger textures to be handled before scanline coherency starts to break down.

[33] Logic Op register readback is via the main register only

9…12	Format1	✓	✓	x	This field selects the format of the texel data T4…T7. The options are 0 = A4L4 1 = L8 2 = I8 3 = A8 4 = 332 5 = A8I8 6 = 5551 7 = 565 8 = 4444 9 = 888 10 = 8888 or YUV
13	ColorOrder1	✓	✓	x	This bit selects the color component order of the texel data T4…T7. The two options are: 0 = AGBR 1 = ARGB
14	AlphaMapEnable1	✓	✓	x	This bit, when set, enables the alpha value of texels T4…T7 to be forced to zero based on testing the color values.
15	AlphaMapSense1	✓	✓	x	This bit selects if the alpha value for texels T4…T7 should be set to zero when the colors are in range or out of range. The options are: 0 = Out of range 1 = In range
16	AlphaMapFiltering	✓	✓	x	This bit, when set, will allow the alpha mapped texels (AlphaMapEnable must be set) to cause the fragment to be discarded depending on the comparison of the number of texels to be alpha mapped with the following three limit fields.
17…19	AlphaMapFilterLimit0	✓	✓	x	This field holds the number of alpha mapped texels in the group T0…T3 which must be exceeded for the fragment to be discarded.
20…22	AlphaMapFilterLimit1	✓	✓	x	This field holds the number of alpha mapped texels in the group T4…T7 which must be exceeded for the fragment to be discarded.
23…26	AlphaMapFilterLimit01	✓	✓	x	This field holds the number of alpha mapped texels in the group T0…T7 which must be exceeded for the fragment to be discarded.
27	MultiTexture	✓	✓	x	This bit, when set, prevents the Alpha Map Filtering logic from testing the I4 interpolant and maybe disregarding the alpha map result of T0…T3 or T4…T7 . This bit should be set for multi texture operation when alpha map filtering is required. It should be clear otherwise.
28	ForceAlphaToOne0	✓	✓	x	This bit, when set, will force the alpha channel of T0…T3 to be set to 1.0 (255) regardless of the color format or the presence of a real alpha channel.
29	ForceAlphaToOne1	✓	✓	x	This bit, when set, will force the alpha channel of T4…T7 to be set to 1.0 (255) regardless of the color format or the presence of a real alpha channel.

30	Shift0				This bit, when set, causes the conversion of T0…T3 for color components less than 8 bits wide to be done by a shift operation, otherwise a scale operation is needed. The shift operation is useful where the exact color (after dithering) is to be preserved for flat shaded areas, such as in a stretch blit.
31	Shift1				This bit, when set, causes the conversion of T4…T7 for color components less than 8 bits wide to be done by a shift operation, otherwise a scale operation is needed. The shift operation is useful where the exact color (after dithering) is to be preserved for flat shaded areas, such as in a stretch blit.

Notes: The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

TextureIndexMode0
TextureIndexMode0And
TextureIndexMode0Or

Name	Type	Offset	Format
TextureIndexMode0	Texture	0xB338	Bitfield
TextureIndexMode0And	Texture	0xB3C0	Bitfield Logic Mask
TextureIndexMode0Or	Texture	0xB3C8	Bitfield Logic Mask
	Control registers		

Bits	Name	Read [34]	Write	Reset	Description
0	Enable	✓	✓	x	When set causes the output to be calculated as defined by the fields in this register, otherwise the fragment's index and interpolation data is set to zero.
1…4	Width	✓	✓	x	This field holds the width of the map as a power of two. The legal range of values for this field is 0 (map width = 1) to 11 (map width = 2048).
5…8	Height	✓	✓	x	This field holds the height of the map as a power of two. The legal range of values for this field is 0 (map width = 1) to 11 (map width = 2048).
9	Border	✓	✓	x	This bit, when set indicates there is a one texel border surrounding the texture map.
10…11	WrapU	✓	✓	x	This field selects how the u coordinate is to be wrapped to fit on the texture map. The options are: 0 = Clamp 1 = Repeat 2 = Mirror 3 = ClampEdge
12…13	WrapV	✓	✓	x	This field selects how the v coordinate is to be wrapped to fit on the texture map. The options are: 0 = Clamp 1 = Repeat 2 = Mirror 3 = ClampEdge
14	MapType	✓	✓	x	This bit selects the type of texture map. The options are 0 = 1D 1 = 2D
15	MagnificationFilter	✓	✓	x	This field selects the magnification filter to use. The options are 0 = Nearest 1 = Linear

[34] Logic Op register readback is via the main register only

16…18	MinificationFilter	✓	✓	x	This field selects the minification filter to use. The options are 0 = Nearest 1 = Linear 2 = NearestMipNearest 3 = NearestMipLinear 4 = LinearMipNearest 5 = LinearMipLinear This field only has an effect when Texture3DEnable or MipMapEnable are true.
19	Texture3DEnable	✓	✓	x	This bit, when set, enables 3D texture index generation.
20	MipMapEnable	✓	✓	x	This bit, when set, enables mip map index generation.
21…22	NearestBias	✓	✓	x	This field defines the bias to add to the u and or v coordinates (after the map's width and height have been taken into account) for nearest neighbour filtering. This can be used to move the texel sample point. The options are: 0 = -0.5 1 = 0 *Use this for OpenGL* 2 = +0.5
23…24	LinearBias	✓	✓	x	This field defines the bias to add to the u and or v coordinates (after the map's width and height have been taken into account) for linear filtering. This can be used to move the texel sample point. The options are: 0 = -0.5 *Use this for OpenGL* 1 = 0 2 = +0.5
25	SourceTexelEnable	✓	✓	x	When set this bit causes the calculated index (i0, j0) to be passed to the Framebuffer Read Unit to be used as a source pixel coordinates. This allows the framebuffer to do stretch blits, rotates, etc.
26…31	Reserved	0	0	x	

Notes: The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

TextureIndexMode1
TextureIndexMode1And
TextureIndexMode1Or

Name	Type	Offset	Format
TextureIndexMode1	Texture	0xB340	Bitfield
TextureIndexMode1And	Texture	0xB3D0	Bitfield Logic Mask
TextureIndexMode1Or	Texture	0xB3D8	Bitfield Logic Mask
	Control registers		

Bits	Name	Read [35]	Write	Reset	Description
0	Enable	✓	✓	x	When set causes the output to be calculated as defined by the fields in this register, otherwise the fragment's index and interpolation data is set to zero.
1…4	Width	✓	✓	x	This field holds the width of the map as a power of two. The legal range of values for this field is 0 (map width = 1) to 11 (map width = 2048).
5…8	Height	✓	✓	x	This field holds the height of the map as a power of two. The legal range of values for this field is 0 (map width = 1) to 11 (map width = 2048).
9	Border	✓	✓	x	This bit, when set indicates there is a one texel border surrounding the texture map.
10…11	WrapU	✓	✓	x	This field selects how the u coordinate is to be wrapped to fit on the texture map. The options are: 0 = Clamp 1 = Repeat 2 = Mirror 3 = ClampEdge
12…13	WrapV	✓	✓	x	This field selects how the v coordinate is to be wrapped to fit on the texture map. The options are: 0 = Clamp 1 = Repeat 2 = Mirror 3 = ClampEdge
14	MapType	✓	✓	x	This bit selects the type of texture map. The options are 0 = 1D 1 = 2D
15	MagnificationFilter	✓	✓	x	This field selects the magnification filter to use. The options are 0 = Nearest 1 = Linear

[35] Logic Op register readback is via the main register only

16…18	MinificationFilter	✓	✓	x	This field selects the minification filter to use. The options are 0 = Nearest 1 = Linear 2 = NearestMipNearest 3 = NearestMipLinear 4 = LinearMipNearest 5 = LinearMipLinear This field only has an effect when Texture3DEnable or MipMapEnable are true.
19	Reserved	0	0	x	
20	MipMapEnable	✓	✓	x	This bit, when set, enables mip map index generation.
21…22	NearestBias	✓	✓	x	This field defines the bias to add to the u and or v coordinates (after the map's width and height have been taken into account) for nearest neighbour filtering. This can be used to move the texel sample point. The options are: 0 = -0.5 1 = 0 *Use this for OpenGL* 2 = +0.5
23…24	LinearBias	✓	✓	x	This field defines the bias to add to the u and or v coordinates (after the map's width and height have been taken into account) for linear filtering. This can be used to move the texel sample point. The options are: 0 = -0.5 *Use this for OpenGL* 1 = 0 2 = +0.5
25	SourceTexelEnable	✓	✓	x	When set this bit causes the calculated index (i0, j0) to be passed to the Framebuffer Read Unit to be used as a source pixel coordinates. This allows the framebuffer to do stretch blits, rotates, etc.
26…31	Reserved	0	0	x	

Notes: The logic operator equivalents behave the same way but the new mode is AND’d or OR’d with the former mode before replacing it.

TextureLodBiasS

Name	**Type**	**Offset**	**Format**
TextureLodBiasS	Texture *Control register*	0x8450	Fixed point

Bits	Name	Read	Write	Reset	Description
0…7	Fraction	✓	✓	x	
8…12	Integer	✓	✓	x	
12…31	Reserved	0	0	x	

Notes: This register holds the 2's complement bias value in 5.8 fixed point format for the S components in the level of detail calculation. Its default value should be zero

TextureLodBiasT

Name	**Type**	**Offset**	**Format**
TextureLodBiasT	Texture *Control register*	0x8458	Fixed point

Bits	Name	Read	Write	Reset	Description
0…7	Fraction	✓	✓	x	
8…12	Integer	✓	✓	x	
12…31	Reserved	0	0	x	

Notes: This register holds the 2's complement bias value in 5.8 fixed point format for the T components in the level of detail calculation. Its default value should be zero

TextureLODScale

Name	**Type**	**Offset**	**Format**
TextureLODScale	Texture *Control register*	0x9340	Float

Bits	Name	Read	Write	Reset	Description
0…31	Scale values	✓	✓	x	32 bit floating point

Notes: Holds the scale values used when calculating the level of detail for a whole triangle. IEEE single precision floating point value

TextureLODScale1

Name	Type	Offset	Format
TextureLODScale1	Texture *Control register*	0x9348	Float

Bits	Name	Read	Write	Reset	Description
0…31	Scale values	✓	✓	x	32 bit floating point

Notes: Holds the scale values used when calculating the level of detail for a whole triangle. IEEE single precision floating point value

TextureMapSize

Name	Type	Offset	Format
TextureMapSize	Texture *Control register*	0xB428	Integer

Bits	Name	Read	Write	Reset	Description
0…23	Offset	✓	✓	x	24 bit unsigned integer
24…31	Reserved	0	0	x	

Notes: This register holds the texel offset between adjacent 2D slices in a 3D texture map. It is a 24 bit unsigned number.

TextureMapWidth[0…15]

Name	Type	Offset	Format
TextureMapWidth[0…15]	Texture *Control register*	0x8580	Bitfield

Bits	Name	Read	Write	Reset	Description
0…11	Width	✓	✓	0	Width (excluding any border)
12	Border enable	✓	✓	0	Border present, if set
13…14	Layout	✓	✓	0	Layout
15	Host Texture	✓	✓	0	HostTexture enabled if set

Notes: These registers hold the width, border, layout and memory type for of each mip map level:

- The width is normally the power of 2 width corresponding to the level, but can be any value in the range 0…4095.
- If a border is present then all mip levels should have the bit set.
- The layout field selects the layout of the texel data in memory for the texture map using *TextureBaseAddr0* register. The options are:
  - 0 = Linear
  - 1 = Patch64 Color buffer
  - 2 = Patch32_2 Large texture maps
  - 3 = Patch2 Small texture maps
- The HostTexture bit is only used if the texture is a physical texture. Logical textures use a bit in the Logical Page Table to identify if a texture is a Host Texture.

TextureReadMode0
TextureReadMode0And
TextureReadMode0Or

Name	**Type**	**Offset**	**Format**
TextureReadMode0	Texture	0xB400	Bitfield
TextureReadMode0And	Texture	0xAC30	Bitfield Logic Mask
TextureReadMode0Or	Texture	0xAC38	Bitfield Logic Mask
	Control registers		

Bits	Name	Read [36]	Write	Reset	Description
0	Enable	✓	✓	x	When set causes any texels needed by the fragment to be read. This is also qualified by the TextureEnable bit in the *Render* command.
1…4	Width	✓	✓	x	This field holds the width of the map as a power of two. The legal range of values for this field is 0 (map width = 1) to 11 (map width = 2048). This is only used when Texture3D is enabled and then is only used for cache management purposes and *not* for address calculations.
5…8	Height	✓	✓	x	This field holds the height of the map as a power of two. The legal range of values for this field is 0 (map height = 1) to 11 (map height = 2048). This is only used when Texture3D is enabled and then is only used for cache management purposes and *not* for address calculations.
9…10	TexelSize	✓	✓	x	This field holds the size of the texels in the texture map. The options are: 0 = 8 bits 1 = 16 bits 2 = 32 bits 3 = 64 bits (Only valid for spans)
11	Textue3D	✓	✓	x	This bit, when set, enables 3D texture index generation. The CombinedCache mode bit should not be set when 3D textures are being used.
12	Combine Caches	✓	✓	x	This bit, when set, causes the two banks of the Primary Cache to be joined together, thereby increasing the size of a single texture map which can be efficiently handled.
13…16	MapBaseLevel	✓	✓	x	This field defines which TextureBaseAddr register should be used to hold the address for map level 0 when mip mapping or the texture map when not mip mapping. Successive map levels are at increasing TextureBaseAddr registers upto (and including) the MaxMaxLevel (next field). 3D textures always use TextureBaseAddr0.

[36] Logic Op register readback is via the main register only

17…20	MapMaxLevel	✓	✓	x	This field defines the maximum TextureBaseAddr register this texture should use when mip mapping. Any attempt to use beyond this level will clamp to this level.
21	LogicalTexture	✓	✓	x	This bit, when set, defines this texture or all mip map levels, if mip mapping, to be logically mapped so undergo logical to physical translation of the texture addresses.
22	Origin	✓	✓	x	This field selects where the origin is for a texture map with a Linear or Patch64 layout. The options are: 0 = Top Left. 1 = Bottom Left A Patch32_2 or Patch2 texture map is always bottom left origin.
23…24	TextureType	✓	✓	x	This field defines any special processing needed on the texel data before it can be used. The options are: 0 = Normal. 1 = Eight bit indexed texture. 2 = Sixteen bit YVYU texture in 422 format. 3 = Sixteen bit VYUY texture in 422 format..
25…27	ByteSwap	✓	✓	x	This field defines the byte swapping, if any, to be done on texel data when it is used as a bitmap. This is automatically done when spans are used. Bit 27, when set, causes adjacent bytes to be swapped, bit 26 adjacent 16 bit words to be swapped and bit 27 adjacent 32 bit words to be swapped. In combination this byte swap the input (ABCDEFGH) as follows: 0 ABCDEFGH 1 BADCFEHG 2 CDABGHEF 3 ABCDEFGH 4 EFGHABCD 5 FEHGBADC 6 GHEFCDAB 7 HGFEDCBA
28	Mirror	✓	✓	x	This bit, when set will mirror any bitmap data. This only works for spans.
29	Invert	✓	✓	x	This bit, when set will invert any bitmap data. This only works for spans.
30	OpaqueSpan	✓	✓	x	This bit, when set, will cause the Span color mask to be modified rather than the pixel mask
31	Reserved	0	0	x	

Notes: The unit is controlled by the *TextureReadMode0* and *TextureReadMode1* registers for texture 0 and texture 1 respectively. Not all combinations of modes across both registers are supported and where there is a clash the modes in *TextureReadMode0* take priority. For per pixel mip mapping the *TextureRead0* and *TextureReadMode1* register should be set up the same as should the *TextureMapWidth0* and *TextureMapWidth1* registers.

N.B. The layout and use of the *TextureReadMode* register is not compatible with GLINT MX: 1, 2, and 4 bit textures are no longer supported.

The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

TextureReadMode1
TextureReadMode1And
TextureReadMode1Or

Name	Type	Offset	Format
TextureReadMode1	Texture	0xB408	Bitfield
TextureReadMode1And	Texture	0xAD40	Bitfield Logic Mask
TextureReadMode1Or	Texture	0xAD48	Bitfield Logic Mask
	Control registers		

Bits	Name	Read[37]	Write	Reset	Description
0	Enable	✓	✓	x	When set causes any texels needed by the fragment to be read. This is also qualified by the TextureEnable bit in the *Render* command.
1…8	Reserved	✓	×	x	
9…10	TexelSize	✓	✓	x	This field holds the size of the texels in the texture map. The options are: 0 = 8 bits 1 = 16 bits 2 = 32 bits 3 = 64 bits (Only valid for spans)
11, 12	Reserved	✓	×	x	
13…16	MapBaseLevel	✓	✓	x	This field defines which TextureBaseAddr register should be used to hold the address for map level 0 when mip mapping or the texture map when not mip mapping. Successive map levels are at increasing TextureBase registers upto (and including) the MapMaxLevel (next field). 3D textures always use TextureBaseAddr0.
17…20	MapMaxLevel	✓	✓	x	This field defines the maximum TextureBaseAddr register this texture should use when mip mapping. Any attempt to use beyond this level will clamp to this level.
21	LogicalTexture	✓	✓	x	This bit, when set, defines this texture or all mip map levels, if mip mapping, to be logically mapped so undergo logical to physical translation of the texture addresses.
22	Origin	✓	✓	x	This field selects where the origin is for a texture map with a Linear or Patch64 layout. The options are: 0 = Top Left. 1 = Bottom Left A Patch32_2 or Patch2 texture map is always bottom left origin.

[37] Logic Op register readback is via the main register only

23…24	TextureType	✓	✓	x	This field defines any special processing needed on the texel data before it can be used. The options are: 0 = Normal. 1 = Eight bit indexed texture. 2 = Sixteen bit YVYU texture in 422 format. 3 = Sixteen bit VYUY texture in 422 format.
25…27	ByteSwap	✓	✓	x	This field defines the byte swapping, if any, to be done on texel data when it is used as a bitmap. This is automatically done when spans are used. Bit 27, when set, causes adjacent bytes to be swapped, bit 26 adjacent 16 bit words to be swapped and bit 27 adjacent 32 bit words to be swapped. In combination this byte swap the input (ABCDEFGH) as follows: 0 ABCDEFGH 1 BADCFEHG 2 CDABGHEF 3 ABCDEFGH 4 EFGHABCD 5 FEHGBADC 6 GHEFCDAB 7 HGFEDCBA
28	Mirror	✓	✓	x	This bit, when set, mirrors any bitmap data. This only works for spans.
29	Invert	✓	✓	x	This bit, when set, inverts any bitmap data. This only works for spans.
30	OpaqueSpan	✓	✓	x	This bit, when set, causes the Span color mask to be modified rather than the pixel mask.
31	Reserved	0	0	x	

Notes: Texture reading is controlled by the *TextureReadMode0* and *TextureReadMode1* registers for texture 0 and texture 1 respectively. Not all combinations of modes across both registers are supported and where there is a clash the modes in TextureReadMode0 take priority. For per pixel mip mapping the *TextureRead0* and *TextureReadMode1* register should be set up the same as should the *TextureMapWidth0* and *TextureMapWidth1* registers.

Note: The layout and use of the *TextureReadMode* register is not compatible with GLINT MX: 1, 2, and 4 bit textures are no longer supported.

The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

TouchLogicalPage

Name	Type	Offset	Format
TouchLogicalPage	Texture *Command*	0xB370	Bitfield

Bits	Name	Read	Write	Reset	Description
0…15	logical page	✓	✓	x	The first Logical Page to mark as stale
15…29	count	✓	✓	x	The number of pages to mark as stale.
30…31	mode	✓	✓	x	0 = Make page(s) non resident 1 = Load page(s) unconditionally. 2 = Make page(s) non resident 3 = Touch page(s) and load if not resident

Notes: This command can be used to touch or mark as non resident a range of pages in the Logical Page Table.

This is useful for preloading and when editing texture maps. For preloading, the command allows you to preload only non-resident pages (mode 3). When editing, the command allows you to mark pages as stale without immediately reloading by setting the mode to “non resident” (mode 2).

TStart

Name	Type	Offset	Format
TStart	Texture *Control register*	0x83A0	Fixed point

Bits	Name	Read	Write	Reset	Description
0…n	Fraction	✓	✓	x	
n…31	Integer	✓	✓	x	

Notes: Initial T value for texture map. The format is 32 bit 2's complement fixed point numbers. The binary point is at an arbitrary location but must be consistent for all S, T and Q values.

UpdateLineStippleCounters

Name	**Type**	**Offset**	**Format**
UpdateLineStippleCounters	Stipple *Command*	0x81B8	Bitfield

Bits	Name	Read	Write	Reset	Description
0	Update Counters Control	✓	✓	x	0=reset counters to 0 1=load from segment register.
1…31	Reserved	0	0	x	.

Notes: This *Command* updates the current line stipple counters: If bit 0 is zero then the counters are set to zero, otherwise they are loaded from the segment register. Useful in drawing stippled wide lines.

UpdateLogicalTextureInfo

Name	**Type**	**Offset**	**Format**
UpdateLogicalTextureInfo	Texture *Command*	0xB368	Tag

Bits	Name	Read	Write	Reset	Description
0…31	Reserved	0	0	x	

Notes: This command updates the Logical Texture Page Table at the page previously set up in the SetLogicalPageInfo command. After the update has been done the logical page number is incremented. The Resident bit is cleared and the Length, MemoryPool, VirtualHostPage and HostPage are set up.

V0FloatR
V0FloatG
V0FloatB
V0FloatA
V0FloatF
V0FloatX
V0FloatY
V0FloatZ

Name	**Type**	**Offset**	**Format**
V0FloatR	Delta	0x91A8	Float
V0FloatG	Delta	0x91B0	Float
V0FloatB	Delta	0x91B8	Float
V0FloatA	Delta	0x91C0	Float
V0FloatF	Delta	0x91C8	Float
V0FloatX	Delta	0x91D0	Float
V0FloatY	Delta	0x91D8	Float
V0FloatZ	Delta	0x91E0	Float
	Control registers		

Bits	Name	Read	Write	Reset	Description
0…31		✓	✓	x	Vertex RGB color, alpha, fog, X, Y and depth

Notes: The R, G, B, Alpha, Fog, X, Y coordinates and Depth values for vertex 0 as IEEE single-precision floating point numbers.

V0FloatKdR
V0FloatKdG
V0FloatKdB

Name	Type	Offset	Format
V0FloatKdR	Delta	0x9068	Float
V0FloatKdG	Delta	0x9070	Float
V0FloatKdB	Delta	0x9078	Float
	Control registers		

Bits	Name	Read	Write	Reset	Description
0…31	Diffuse	✓	✓	x	Vertex diffuse texture value

Notes: The diffuse KdR, G and B texture values for vertex 0 as IEEE single-precision floating point numbers.

V0FloatKsR
V0FloatKsG
V0FloatKsB

Name	Type	Offset	Format
V0FloatKsR	Delta	0x9050	Float
V0FloatKsG	Delta	0x9058	Float
V0FloatKsB	Delta	0x9060	Float
	Control registers		

Bits	Name	Read	Write	Reset	Description
0…31	Specular	✓	✓	x	Vertex specular texture value

Notes: The specular KsR, G and B texture values for vertex 0 as IEEE single-precision floating point numbers.

V0FloatPackedColor

Name	Type	Offset	Format
V0FloatPackedColor	Delta *Control register*	0x91F0	Bitfield

Bits	Name	Read	Write	Reset	Description
0…7	R	0	✓	x	
8…15	G	0	✓	x	
16…23	B	0	✓	x	
24…31	A	0	✓	x	

Notes: Vertex 0 color definition - the packed color registers hold the red, green, blue and alpha components in the same 32 bit word. When written to, the components are separated, converted to floating point format, and loaded into the registers. The color order in the registers is set by bit 18 in the *DeltaMode* register:

Bit31… *Bit0*
0 = Alpha (or Fog), Blue, Green, Red
1 = Alpha (or Fog), Red, Green, Blue

Reading back from the packed color registers returns zero.

V0FloatPackedDiffuse

Name	Type	Offset	Format
V0FloatPackedDiffuse	Delta *Control register*	0x9048	Bitfield

Bits	Name	Read	Write	Reset	Description
0…7	R	0	✓	x	
8…15	G	0	✓	x	
16…23	B	0	✓	x	
24…31	A	0	✓	x	

Notes: The color order in the registers is set by bit 18 in the *DeltaMode* register:

Bit31… *Bit0*
0 = Alpha (or Fog), Blue, Green, Red
1 = Alpha (or Fog), Red, Green, Blue

Reading back from the packed color registers returns zero.

V0FloatPackedSpecularFog

Name	Type	Offset	Format
V0FloatPackedSpecularFog	Delta *Control register*	0x91F8	Bitfield

Bits	Name	Read	Write	Reset	Description
0…7	R	0	✓	x	
8…15	G	0	✓	x	
16…23	B	0	✓	x	
24…31	Fog	0	✓	x	

Notes: Vertex 0 specular definition - packed specular registers are treated in the same way as packed color registers: the RGB components are separated, converted to the internal floating point format, and loaded into the registers. When loaded from a packed register, the specular range is 0 to 1.0. The A component is converted into an internal format and loaded into the fog register - when loaded from the packed register, the fog range is 0 to 1.0.

The color order in the registers is set by bit 18 in the *DeltaMode* register:

Bit31… *Bit0*
0 = Alpha (or Fog), Blue, Green, Red
1 = Alpha (or Fog), Red, Green, Blue

Reading back from the packed color registers returns zero.

V0FloatS
V0FloatT
V0FloatQ

Name	Type	Offset	Format
V0FloatS	Delta	0x9180	Float
V0FloatT	Delta	0x9188	Float
V0FloatQ	Delta	0x9190	Float
	Control registers		

Bits	Name	Read	Write	Reset	Description
0…31	Texture	✓	✓	x	Vertex texture values

Notes: The texture S, T and Q values for vertex 0 as IEEE single-precision floating point numbers.

V0FloatS1
V0FloatT1
V0FloatQ1

Name	Type	Offset	Format
V0FloatS1	Delta	0x9000	Float
V0FloatT1	Delta	0x9008	Float
V0FloatQ1	Delta	0x9010	Float
	Control registers		

Bits	Name	Read	Write	Reset	Description
0…31	Texture	✓	✓	x	Vertex texture value

Notes: The texture S1, T1 and Q1 values for vertex 0 as IEEE single-precision floating point numbers.

V1FloatR
V1FloatG
V1FloatB
V1FloatA
V1FloatF
V1FloatX
V1FloatY
V1FloatZ

Name	Type	Offset	Format
V1FloatR	Delta	0x9228	Float
V1FloatG	Delta	0x9230	Float
V1FloatB	Delta	0x9238	Float
V1FloatA	Delta	0x9240	Float
V1FloatF	Delta	0x9248	Float
V1FloatX	Delta	0x9250	Float
V1FloatY	Delta	0x9258	Float
V1FloatZ	Delta	0x9260	Float
	Control registers		

Bits	Name	Read	Write	Reset	Description
0…31		✓	✓	x	Vertex RGB color, alpha, fog, X, Y and depth

Notes: The R, G, B, Alpha, Fog, X, Y coordinates and Depth values for vertex 1 as IEEE single-precision floating point numbers.

V1FloatKdR
V1FloatKdG
V1FloatKdB

Name	Type	Offset	Format
V1FloatKdR	Delta	0x90E8	Float
V1FloatKdG	Delta	0x90F0	Float
V1FloatKdB	Delta	0x90F8	Float
	Control registers		

Bits	Name	Read	Write	Reset	Description
0…31	Diffuse	✓	✓	x	Vertex diffuse texture values

Notes: The diffuse KdR, G and B texture values for vertex 1 as IEEE single-precision floating point numbers.

V1FloatKsR
V1FloatKsG
V1FloatKsB

Name	Type	Offset	Format
V1FloatKsR	Delta	0x90D0	Float
V1FloatKsG	Delta	0x90D8	Float
V1FloatKsB	Delta	0x90E0	Float
	Control registers		

Bits	Name	Read	Write	Reset	Description
0…31	Diffuse	✓	✓	x	Vertex diffuse texture value

Notes: The diffuse KdR, G and B texture values for vertex 1 as IEEE single-precision floating point numbers.

V1FloatPackedColor

Name	Type	Offset	Format
V1FloatPackedColor	Delta	0x9270	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	R	0	✓	x	
8…15	G	0	✓	x	
16…23	B	0	✓	x	
24…31	Fog	0	✓	x	

Notes: Vertex 1 color definition - the packed color registers hold the red, green, blue and alpha components in the same 32 bit word. When written to, the components are separated, converted to the internal floating point format, and loaded into the registers. The color order in the registers is set by bit 18 in the *DeltaMode* register:

Bit31… *Bit0*
0 = Alpha (or Fog), Blue, Green, Red
1 = Alpha (or Fog), Red, Green, Blue

Reading back from the packed color registers returns zero.

V1FloatPackedDiffuse

Name	Type	Offset	Format
V1FloatPackedDiffuse	Delta *Control register*	0x90C8	Bitfield

Bits	Name	Read	Write	Reset	Description
0…7	R	0	✓	x	
8…15	G	0	✓	x	
16…23	B	0	✓	x	
24…31	A	0	✓	x	

Notes: The color order in the registers is set by bit 18 in the *DeltaMode* register:
Bit31… *Bit0*
0 = Alpha (or Fog), Blue, Green, Red
1 = Alpha (or Fog), Red, Green, Blue
Reading back from the packed color registers returns zero.

V1FloatPackedSpecularFog

Name	Type	Offset	Format
V1FloatPackedSpecularFog	Delta *Control register*	0x9278	Bitfield

Bits	Name	Read	Write	Reset	Description
0…7	R	0	✓	x	
8…15	G	0	✓	x	
16…23	B	0	✓	x	
24…31	A	0	✓	x	

Notes: Vertex 1 specular definition - packed specular registers are treated in the same way as packed color registers: the RGB components are separated, converted to the internal floating point format, and loaded into the registers. When loaded from a packed register, the specular range is 0 to 1.0. The A component is converted into an internal format and loaded into the fog register - when loaded from the packed register, the fog range is 0 to 1.0.

The color order in the registers is set by bit 18 in the *DeltaMode* register:
Bit31… *Bit0*
0 = Alpha (or Fog), Blue, Green, Red
1 = Alpha (or Fog), Red, Green, Blue
Reading back from the packed color registers returns zero.

V1FloatS
V1FloatT
V1FloatQ

Name	Type	Offset	Format
V1FloatS	Delta	0x9200	Float
V1FloatT	Delta	0x9208	Float
V1FloatQ	Delta	0x9210	Float
	Control registers		

Bits	Name	Read	Write	Reset	Description
0…31	Texture	✓	✓	x	Vertex texture values

Notes: The texture S, T and Q values for vertex 1 as IEEE single-precision floating point numbers.

V1FloatS1
V1FloatT1
V1FloatQ1

Name	Type	Offset	Format
V1FloatS1	Delta	0x9080	Float
V1FloatT1	Delta	0x9088	Float
V1FloatQ1	Delta	0x9090	Float
	Control registers		

Bits	Name	Read	Write	Reset	Description
0…31	Texture	✓	✓	x	Vertex texture values

Notes: The texture S1, T1 and Q1 values for vertex 1 as IEEE single-precision floating point numbers.

V2FloatR
V2FloatG
V2FloatB
V2FloatA
V2FloatF
V2FloatX
V2FloatY
V2FloatZ

Name	Type	Offset	Format
V2FloatR	Delta	0x92A8	Float
V2FloatG	Delta	0x92B0	Float
V2FloatB	Delta	0x92B8	Float
V2FloatA	Delta	0x92C0	Float
V2FloatF	Delta	0x92C8	Float
V2FloatX	Delta	0x92D0	Float
V2FloatY	Delta	0x92D8	Float
V2FloatZ	Delta	0x92E0	Float
	Control registers		

Bits	Name	Read	Write	Reset	Description
0…31		✓	✓	x	Vertex RGB color, alpha, fog, X, Y and depth

Notes: The R, G, B, Alpha, Fog, X, Y coordinates and Depth values for vertex 2 as IEEE single-precision floating point numbers.

V2FloatKdR
V2FloatKdG
V2FloatKdB

Name	Type	Offset	Format
V2FloatKdR	Delta	0x9168	Float
V2FloatKdG	Delta	0x9170	Float
V2FloatKdB	Delta	0x9178	Float
	Control registers		

Bits	Name	Read	Write	Reset	Description
0…31	Diffuse	✓	✓	x	Vertex diffuse texture values

Notes: The diffuse KdR, G and B texture values for vertex 2 as IEEE single-precision floating point numbers.

V2FloatKsR
V2FloatKsG
V2FloatKsB

Name	Type	Offset	Format
V2FloatKsR	Delta	0x9150	Float
V2FloatKsG	Delta	0x9158	Float
V2FloatKsB	Delta	0x9160	Float
	Control registers		

Bits	Name	Read	Write	Reset	Description
0…31	Diffuse	✓	✓	x	Vertex diffuse texture values

Notes: The specular KsR, G and B texture values for vertex 2 as IEEE single-precision floating point numbers.

V2FloatPackedColor

Name	Type	Offset	Format
V2FloatPackedColor	Delta	0x92F0	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	R	0	✓	x	
8…15	G	0	✓	x	
16…23	B	0	✓	x	
24…31	A	0	✓	x	

Notes: Vertex 2 color definition - the packed color registers hold the red, green, blue and alpha components in the same 32 bit word. When written to, the components are separated, converted to an internal format, and loaded into the registers. The color order in the registers is set by bit 18 in the *DeltaMode* register:

Bit31… *Bit0*

0 = Alpha (or Fog), Blue, Green, Red

1 = Alpha (or Fog), Red, Green, Blue

Reading back from the packed color registers returns zero.

V2FloatPackedDiffuse

Name	Type	Offset	Format
V2FloatPackedDiffuse	Delta *Control register*	0x9148	Bitfield

Bits	Name	Read	Write	Reset	Description
0…7	R	0	✓	x	
8…15	G	0	✓	x	
16…23	B	0	✓	x	
24…31	A	0	✓	x	

Notes: The color order in the registers is set by bit 18 in the *DeltaMode* register:

Bit31… *Bit0*
0 = Alpha (or Fog), Blue, Green, Red
1 = Alpha (or Fog), Red, Green, Blue

Reading back from the packed color registers returns zero.

V2FloatPackedSpecularFog

Name	Type	Offset	Format
V2FloatPackedSpecularFog	Delta *Control register*	0x92F8	Bitfield

Bits	Name	Read	Write	Reset	Description
0…7	R	0	✓	x	
8…15	G	0	✓	x	
16…23	B	0	✓	x	
24…31	A	0	✓	x	

Notes: Vertex 2 specular definition - packed specular registers are treated in the same way as packed color registers: the RGB components are separated, converted to the internal floating point format, and loaded into the registers. When loaded from a packed register, the specular range is 0 to 1.0. The A component is converted into an internal format and loaded into the fog register - when loaded from the packed register, the fog range is 0 to 1.0.

The color order in the registers is set by bit 18 in the *DeltaMode* register:

Bit31… *Bit0*
0 = Alpha (or Fog), Blue, Green, Red
1 = Alpha (or Fog), Red, Green, Blue

Reading back from the packed color registers returns zero.

V2FloatS
V2FloatT
V2FloatQ

Name	**Type**	**Offset**	**Format**
V2FloatS	Delta	0x9280	Float
V2FloatT	Delta	0x9288	Float
V2FloatQ	Delta	0x9290	Float
	Control registers		

Bits	Name	Read	Write	Reset	Description
0…31	Texture	✓	✓	x	Vertex texture values

Notes: The texture S, T and Q values for vertex 2 as IEEE single-precision floating point numbers.

V2FloatS1
V2FloatT1
V2FloatQ1

Name	**Type**	**Offset**	**Format**
V2FloatS1	Delta	0x9100	Float
V2FloatT1	Delta	0x9108	Float
V2FloatQ1	Delta	0x9110	Float
	Control registers		

Bits	Name	Read	Write	Reset	Description
0…31	Texture	✓	✓	x	Vertex texture values

Notes: The texture S1, T1 and Q1 values for vertex 2 as IEEE single-precision floating point numbers.

Vertex0

Name	**Type**	**Offset**	**Format**
Vertex0	Input *Control register*	0xB7B8	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Index	✓	✓	x	Index into Vertex buffer

Notes: The vertex data can be loaded without using one of the primitive types using the Vertex0, Vertex1, and Vertex2 registers. These registers specify the vertex store to load, and the data field holds the index into the array.

Vertex1

Name	**Type**	**Offset**	**Format**
Vertex1	Input *Control register*	0xB7C0	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Vertex	✓	✓	x	Index into Vertex buffer

Notes: The vertex data can be loaded without using one of the primitive types using the Vertex0, Vertex1, and Vertex2 registers. These registers specify the vertex store to load, and the data field holds the index into the array.

Vertex2

Name	Type	Offset	Format
Vertex2	Input *Control register*	0xB7C8	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Index	✓	✓	x	Index into Vertex buffer

Notes: The vertex data can be loaded without using one of the primitive types using the Vertex0, Vertex1, and Vertex2 registers. These registers specify the vertex store to load, and the data field holds the index into the array.

VertexBaseAddress

Name	Type	Offset	Format
VertexBaseAddress	Input *Control register*	0xB708	Integer

Bits	Name	Read	Write	Reset	Description
0…1	Reserved	0	0	x	
2…31	Address	✓	✓	x	32 bit address of base of buffer

Notes:

VertexControl

Name	Type	Offset	Format
VertexControl	Input *Control register*	0xB798	Bitfield

Bits	Name	Read	Write	Reset	Description
0-4	Size	✓	✓	x	Size of vertex in 32 words
5	CacheEnable	✓	✓	x	0 = cache off, 1 = cache on
6	Flat	✓	✓	x	0 = off, 1 = on
7	ReadAll	✓	✓	x	0 = off, 1 = on
8	SkipFlags	✓	✓	x	0 = off, 1 = on
9	OGL	✓	✓	x	0 = D3D, 1 = OGL (used to define provoking vertex behaviour)
10	Line2D	✓	✓	x	0 = off, 1 = 0n
11-31	Reserved	0	0	x	

Notes:

VertexData

Name	Type	Offset	Format
VertexData	Input *Control register*	0xB7E8	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Data	✓	✓	x	Vertex data

Notes: The vertex data can be loaded without using one of the primitive types using the Vertex0, Vertex1, and Vertex2 registers. These registers specify the vertex store to load, and the data field holds the index into the array. The VertexData register is used for inline vertex data.

VertexData0

Name	Type	Offset	Format
VertexData0	Input *Control register*	0xB7D0	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Data	✓	✓	x	Vertex data

Notes:

VertexData1

Name	Type	Offset	Format
VertexData1	Input *Control register*	0xB7D8	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Data	✓	✓	x	Vertex data

Notes:

VertexData2

Name	**Type**	**Offset**	**Format**
VertexData2	Input *Control register*	0xB7E0	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Data	✓	✓	x	Vertex data

Notes:

VertexFormat

Name	**Type**	**Offset**	**Format**
VertexFormat	Input *Control register*	0xB790	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Mask	✓	✓	x	Mask of data valid in vertex

Notes:

VertexLineList

Name	**Type**	**Offset**	**Format**
VertexLineList	Input *Control register*	0xB760	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Count	✕	✓	x	Number of vertices in primitive

Notes:

VertexLineStrip

Name	**Type**	**Offset**	**Format**
VertexLineStrip	Input *Control register*	0xB768	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Count	✕	✓	x	Number of vertices in primitive

Notes:

VertexPointList

Name	**Type**	**Offset**	**Format**
VertexPointList	Input *Control register*	0xB770	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Count	✕	✓	x	Number of vertices in primitive

Notes:

VertexPolygon

Name	**Type**	**Offset**	**Format**
VertexPolygon	Input *Control register*	0xB778	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Count	✕	✓	x	Number of vertices in primitive

Notes:

VertexTagList[0…15]

Name	Type	Offset	Format
VertexTagList[0…15]	Input *Control register*	0xB800	Bitfield

Bits	Name	Read	Write	Reset	Description
0…10	Tag	✓	✓	x	Tag to use for corresponding vertex data item
11…31	Reserved	0	0	x	

Notes: Typical usage would use the TagList to define the order in which data is delivered; the format mask and vertex size are used to set which modes are enabled (so if z is enabled the z bit in the format mask is set and the vertex size increased by 1).

VertexTagList[16…31]

Name	Type	Offset	Format
VertexTagList[16…31]	Input *Control register*	0xB880	Bitfield

Bits	Name	Read	Write	Reset	Description
0…10	Tag	✓	✓	x	Tag to use for corresponding vertex data item
11…31	Reserved	0	0	x	

Notes: Typical usage would use the TagList to define the order in which data is delivered; the format mask and vertex size are used to set which modes are enabled (so if z is enabled the z bit in the format mask is set and the vertex size increased by 1).

VertexTriangleFan

Name	Type	Offset	Format
VertexTriangleFan	Input *Control register*	0xB750	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Count	✗	✓	x	Number of vertices in primitive

Notes:

VertexTriangleList

Name	Type	Offset	Format
VertexTriangleList	Input *Control register*	0xB748	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Count	✗	✓	x	Number of vertices in primitive

Notes:

VertexTriangleStrip

Name	Type	Offset	Format
VertexTriangleStrip	Input *Control register*	0xB750	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Count	✗	✓	x	Number of vertices in primitive

Notes:

VertexValid

Name	**Type**	**Offset**	**Format**
VertexValid	Input *Control register*	0xB788	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Mask	✓	✓	x	Mask of data valid in vertex

Notes:

VTGAddress

Name	**Type**	**Offset**	**Format**
VTGAddress	Framebuffer *Command*	0xB0B0	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Address	✓	✓	x	32 bit value

Notes: The VTG and RAMDAC can be read and written via the PCI bypass, but sometimes it is useful to control them synchronously with core rendering activities. This can be done by using the VTGAddress and VTGData commands. The address is sent first followed by the data. The address and data are the same as would be used if the VTG, Ramdac or any other device on the PCI bypass were accessed via the bypass.

The core does not interpret the data in any way and is just the communications path. The VTG data and address is routed via the FB Memory Interface.

VTGData

Name	Type	Offset	Format
VTGAddress	Framebuffer *Command*	0xB0B8	Integer

Bits	Name	Read	Write	Reset	Description
0…31	VTG Data	✓	✓	x	32 bit value

Notes: This register holds the data for the VTG or bypass write and instigates the action via the FB Memory Controller.

The VTG and RAMDAC can be read and written via the PCI bypass, but sometimes it is useful to control them synchronously with core rendering activities. This can be done by using the VTGAddress and VTGData commands. The address is sent first followed by the data. The address and data are the same as would be used if the VTG, Ramdac or any other device on the PCI bypass were accessed via the bypass.

The core does not interpret the data in any way and is just the communications path. The VTG data and address is routed via the FB Memory Interface.

WaitforCompletion

Name	Type	Offset	Format
WaitforCompletion	Rasterizer *Command*	0x80B8	Bitfield

Bits	Name	Read	Write	Reset	Description
0, 1	Event	0	✓	x	0 = LB Reads and writes and FB reads and writes 1 = LB Reads and FB Reads 2 = RenderSync 3 = ScanlineSyncU
2…31	Unused	0	0	x	

Notes: *Command*: This is used to suspend core graphics processing until outstanding reads and writes in both localbuffer and framebuffer memory have completed, or some other combination of events described above has taken place. This is intended to prevent a new primitive from starting to be rasterized before the previous primitive is completely finished. It would be used, for example, to separate texture downloads from the surrounding primitives.

The same functionality can be achieved using the Sync register and waiting for it in the Host Out FIFO; however, this method doesn't involve the host and can be inserted into a DMA buffer.

Window

Name	Type	Offset	Format
Window	Localbuffer *Control register*	0x8980	Bitfield

Bits	Name	Read	Write	Reset	Description
0…2	Reserved	0	0	x	
3	ForceLB Update	✓	✓	x	This bit, when set, disregards the results of the stencil and depth tests and forced the local buffer to be updated.
4	LBUpdate Source	✓	✓	x	This bit selects the data to be written to the local buffer. The two options are: 0 = LB data. 1 = Registers.
5…8	Reserved	0	0	x	
9...16	FrameCount	✓	✓	x	Reserved
17	Stencil FCP	✓	✓	x	This bit, when set, enables the FCP tests and substitution to occur for the Stencil field.
18	DepthFCP	✓	✓	x	This bit, when set, enables the FCP tests and substitution to occur for the Depth field.
19	OverrideWrite Filtering	✓	✓	x	This bit, when set, prevents writes to the local buffer from being filtered out because this unit has not changed the data.
20…31	Reserved	0	0	x	

Notes: Stencil operation generally is under control of the Window register:

- The Force LB Update bit, when set overrides all the tests done in the Stencil and Depth units and the per unit enables to force the local buffer to be updated. When this bit is clear any update is conditional on the outcome of the stencil and depth tests. The main use of this bit is during window initialisation or copy. It may also be useful for hardware diagnostics.
- The data used during ForceLBUpdate depends on the settings in the LBUpdateSource bit. When this bit is 0 the data is taken from the local buffer. Note that either destination or source local buffer data can be used depending on which is enabled. If both are enabled then the destination local buffer data will be used.
- When the LBUpdateSource bit is set the source of the stencil and depth data is determined by the StencilMode and DepthMode registers respectively.
- The Override Write Filtering control bit, when set causes the testing of LBData = LBWriteData to always fail. This is mainly used when the GID field needs to be changed. It also allows the LBReadFormat to be different to the LBWriteFormat so the write data as seen by the memory is really different to the data that was read.

WindowOrigin

Name	**Type**	**Offset**	**Format**
WindowOrigin	Scissor *Command*	0x81C8	Bitfield

Bits	Name	Read	Write	Reset	Description
0…15	X coordinate				X coordinate as 2's complement number
16…31	Y coordinate				Y coordinate as 2's complement number

Notes: This register holds the window origin. As each fragment is generated by the rasterizer, this origin is added to the coordinates of the fragment to generate its localbuffer coordinate when the depth and stencil buffers are patched.

XBias

Name	**Type**	**Offset**	**Format**
XBias	Delta *Control register*	0x9480	Float

Bits	Name	Read	Write	Reset	Description
0…31	Offset	✓	✓	x	

Notes: This register holds the single precision floating point bias added to the vertices' X coordinate (if enabled) just before rasterization.

YBias

Name	Type	Offset	Format
YBias	Delta *Control register*	0x9488	Float

Bits	Name	Read	Write	Reset	Description
0…31		✓	✓	x	

Notes: This register holds the single precision floating point bias added to the vertices' Y coordinate (if enabled) just before rasterization.

YLimits

Name	Type	Offset	Format
YLimits	Rasterizer *Command*	0x80A8	Bitfield

Bits	Name	Read	Write	Reset	Description
0…15	Ymin	✓	✓	x	2's complement min Y value
16…31	Ymax	✓	✓	x	2’s complement max Y value

Notes: Defines the Y extent the Rasterizer should fill between. A scanline is filled if its Y value satisifies Ymin<Y<Ymax.

YUVMode

Name	Type	Offset	Format
YUVMode	YUV *Control register*	0x8F00	Bitfield

Bits	Name	Read	Write	Reset	Description
0	Enable	✓	✓	x	When set causes the fragment's color values to be converted from YUV to RGB. If this bit is clear then the fragment's color is passed unchanged
1…31	Reserved	0	0	x	

Notes: The conversion goes from the YCbCr color space to RGB. The term YCbCr is used interchangeably with YUV.
The output of the conversion is an RGB triple with each component 8 bits wide. The alpha component is passed through unchanged.

ZBias

Name	Type	Offset	Format
ZBias	Delta *Control register*	0x94F8	Float

Bits	Name	Read	Write	Reset	Description
0…31	Offset	✓	✓	x	

Notes: This register holds the single precision floating point bias added to the vertices' Z coordinate (if enabled) just before rasterization.

ZFogBias

Name	Type	Offset	Format
ZFogBias	Delta *Control register*	0x86B8	Float

Bits	Name	Read	Write	Reset	Description
0…31	Bias	✓	✓	x	2's complement value for Z

Notes: This register holds the 32 bit 2's complement value to add to the Z value extracted from the fog DDA before it is clamped and scaled. The bias essentially is used to set the Z value below which no blending occurs.

ZStartL

Name	Type	Offset	Format
ZStartL	Depth *Control register*	0x89B8	Fixed point pair

Bits	Name	Read	Write	Reset	Description
0…15	Reserved	0	0	x	LSBs all 0
16…31	Integer	✓	✓	x	16bit LSB part of 32.16 fixed point value

Notes: This register holds the lower 16 bits of the 48 bit 2's complement Z start value. These bits are held in bits 16…31 of the data field. With ZstartU, it sets the start value for depth interpolation. ZStartU holds the most significant bits, and ZStartL the least significant bits. The value is in 2's complement 32.16 fixed point format.

ZStartU

Name	Type	Offset	Format
ZStartU	Stencil *Control register*	0x89B0	Fixed point pair

Bits	Name	Read	Write	Reset	Description
0…31	dZdxU	✓	✓	x	32 bit integer

Notes: This register holds the upper 32 bits of the 48 bit 2's complement Z start value.
With ZstartL, it sets the start value for depth interpolation. ZStartU holds the most significant bits, and ZStartL the least significant bits. The value is in 2's complement 32.16 fixed point format.

6

Register Cross Reference

This chapter provides alphabetically- and offset-sorted Region 0 register listings.

6.1 Registers Alphabetically Sorted

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
AALineWidth	✔	✔	Delta	94C0	x	float	✕
AAPointSize	✔	✔	Delta	94A0	x	float	✕
AGPControl	✔	✔	Control Status	0078		bitfield	
AlphaBlendAlphaMode	✔	✔	Alpha blend	AFA8	x	bitfield	✕
AlphaBlendAlphaModeAnd	✕	✔	Alpha blend	AD30	x	bitfield	✕
AlphaBlendAlphaModeOr	✕	✔	Alpha blend	AD38	x	bitfield	✕
AlphaBlendColorMode	✔	✔	Alpha blend	AFA0	x	bitfield	✕
AlphaBlendColorModeAnd	✕	✔	Alpha blend	ACB0	x	bitfield	✕
AlphaBlendColorModeOr	✕	✔	Alpha blend	ACB8	x	bitfield	✕
AlphaDestColor	✔	✔	Alpha blend	AF88	x	bitfield	✕
AlphaSourceColor	✔	✔	Alpha blend	AF80	x	integer	✕
AlphaTestMode	✔	✔	Alpha Blend & Alpha Test	8800	x	bitfield	✕
AlphaTestModeAnd	✕	✔	Alpha Blend & Alpha Test	ABF0	x	bitfield	✕
AlphaTestModeOr	✕	✔	Alpha Blend & Alpha Test	ABF8	x	bitfield	✕
AntialiasMode	✔	✔	Alpha test	8808	x	bitfield	✕

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
AntialiasModeAnd	✗	✔	Alpha test	AC00	x	bitfield	✗
AntialiasModeOr	✗	✔	Alpha test	AC08	x	bitfield	✗
ApertureOne	✔	✔	Control Status	0050		bitfield	
ApertureTwo	✔	✔	Control Status	0058		bitfield	
AreaStippleMode	✔	✔	Stipple	81A0	x	Bitfield	✗
AreaStippleModeAnd	✗	✔	Stipple	ABD0	x	bitfield	✗
AreaStippleModeOr	✗	✔	Stipple	ABD8	x	bitfield	✗
AreaStipplePattern[0…15]	✔	✔	Stipple	8200	x	Bitfield	✗
AreaStipplePattern[16…31]	✔	✔	Stipple	8280	x	Bitfield	✗
AStart	✔	✔	Color DDA	87C8	x	fixed	✗
BackgroundColor	✔	✔	Logic Ops	B0C8	x	integer	✗
BasePageOfWorking Set	✔	✔	Texture Read	B4C8	x	integer	✗
BasePageOfWorking SetHost	✔	✔	Texture Read	B4E0	x	integer	✗
BitMaskPattern	✗	✔	Raster-izer	8068	x	Integer	✔✗
BorderColor0	✔	✔	Texture filter	84A8	x	bitfield	✗
BorderColor1	✔	✔	Texture filter	84F8	x	bitfield	✗
BStart	✔	✔	Color DDA	87B0	x	fixed	✗
ByAperture1Mode	✔	✔	Bypass Control	0300		Bitfield	
ByAperture1Stride	✔	✔	Bypass Control	0308		Integer	
ByAperture1UStart	✔	✔	Bypass Control	0318		Integer	
ByAperture1VStart	✔	✔	Bypass Control	0320		Integer	
ByAperture1YStart	✔	✔	Bypass Control	0310		Integer	
ByAperture2Mode	✔	✔	Bypass Control	0328		Bitfield	
ByAperture2Stride	✔	✔	Bypass Control	0330		Integer	
ByAperture2UStart	✔	✔	Bypass Control	0340		Integer	
ByAperture2VStart	✔	✔	Bypass Control	0348		Integer	
ByAperture2YStart	✔	✔	Bypass Control	0338		Integer	

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
ByDMAReadCommandBase	✔	✔	Bypass Control	0378		Integer	
ByDMAReadCommandCount	✔	✔	Bypass Control	0380		Integer	
ByDMAReadMode	✔	✔	Bypass Control	0350		Bitfield	
ByDMAReadStride	✔	✔	Bypass Control	0358		Integer	
ByDMAReadUStart	✔	✔	Bypass Control	0368		Integer	
ByDMAReadVStart	✔	✔	Bypass Control	0370		Integer	
ByDMAReadYStart	✔	✔	Bypass Control	0360		Integer	
ByDMAWriteCommandBase	✔	✔	Bypass Control	03B0		Integer	
ByDMAWriteCommandCount	✔	✔	Bypass Control	03B8		Integer	
ByDMAWriteMode	✔	✔	Bypass Control	0388		Bitfield	
ByDMAWriteStride	✔	✔	Bypass Control	0390		Integer	
ByDMAWriteUStart	✔	✔	Bypass Control	03A0		Integer	
ByDMAWriteVStart	✔	✔	Bypass Control	03A8		Integer	
ByDMAWriteYStart	✔	✔	Bypass Control	0398		Integer	
ChipConfig	✔	✔	Control Status	0070		bitfield	
ChromaFailColor	✔	✔	Color DDA & Alpha Blend	AF98	x	bitfield	✕
ChromaLower	✔	✔	Color DDA & Alpha Blend	8F10	x	bitfield	✕
ChromaPassColor	✔	✔	Color DDA & Alpha Blend	AF90	x	bitfield	✕
ChromaTestMode	✔	✔	Color DDA & Alpha Blend	8F18	x	bitfield	✕
ChromaTestModeAnd	✕	✔	Color DDA & Alpha Blend	ACC0	x	bitfield	✕

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
ChromaTestModeOr	✕	✔	Color DDA & Alpha Blend	ACC8	x	bitfield	✕
ChromaUpper	✔	✔	Color DDA & Alpha Blend	8F08	x	bitfield	✕
Color	✔	✔		87F0	x	bitfield	✕
ColorDDAMode	✔	✔	Color DDA	87E0	x	bitfield	
ColorDDAModeAnd	✕	✔	Color DDA	ABE0	x	bitfield	✕
ColorDDAModeOr	✕	✔	Color DDA	ABE8	x	bitfield	✕
Command Interrupt	✕	✔	Host In	A990	x	bitfield	✕
Config2D	✕	✔	Global	B618	x	bitfield	✕
ConstantColor	✔	✔	Color DDA	87E8	x	bitfield	
ConstantColorDDA	✕	✔	Color DDA	AFB0	x	bitfield	✕
ContextData	✕	✔	Global	8DD0	x	bitfield	✕
ContextDump	✕	✔	Global	8DC0	x	bitfield	✔
ContextRestore	✕	✔	Global	8DC8	x	bitfield	✔
Continue	✕	✔	Raster-izer	8058	x	Integer	✔
ContinueNewDom	✕	✔	Raster-izer	8048	x	Integer	✔
ContinueNewLine	✕	✔	Raster-izer	8040	x	Integer	✔
ContinueNewSub	✕	✔	Raster-izer	8050	x	Integer	✔
ControlDMAAddress	✔	✔	Control Status	0028		integer	
ControlDMAControl	✔	✔	Control Status	0060		bitfield	
ControlDMACount	✔	✔	Control Status	0030		integer	
Count	✔	✕	Raster-izer	8030	x	Integer	✕
dAdx	✔	✔	Color DDA	87D0	x	fixed	✕
dAdyDom	✔	✔	Color DDA	87D8	x	fixed	
dBdx	✔	✔	Color DDA	87B8	x	fixed	✕
dBdyDom	✔	✔	Color DDA	87C0	x	fixed	✕
DeltaControl	✔	✔	Delta	9350	x	bitfield	✕

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
DeltaControlAnd	✕	✔	Delta	AB20	x	bitfield	✕
DeltaControlOr	✕	✔	Delta	AB28	x	bitfield	✕
DeltaMode	✔	✔	Delta	9300	x	bitfield	✕
DeltaModeAnd	✕	✔	Delta	AAD0	x	bitfield	✕
DeltaModeOr	✕	✔	Delta	AAD8	x	bitfield	✕
Depth	✔	✔	Depth	89A8	x	integer	✔✕
DepthMode	✔	✔	Depth	89A0	x	bitfield	✕
DepthModeAnd	✕	✔	Depth	AC70	x	bitfield	✕
DepthModeOr	✕	✔	Depth	AC78	x	bitfield	✕
dFdx	✔	✔	Fog	86A8	x	fixed	✕
dFdyDom	✔	✔	Fog	86B0	x	fixed	✕
dGdx	✔	✔	Color DDA	87A0	x	fixed	✕
dGdyDom	✔	✔	Color DDA	87A8	x	fixed	✕
DisplayData			Video Control	3068		bitfield	
DitherMode	✔	✔	Dither	8818	x	bitfield	✕
DitherModeAnd	✕	✔	Dither	ACD0	x	bitfield	✕
DitherModeOr	✕	✔	Dither	ACD8	x	bitfield	✕
dKdBdx	✔	✔	Texture	8D38	x	fixed	✕
dKdBdyDom	✔	✔	Texture	8D40	x	fixed	✕
dKdGdx	✔	✔	Texture	8D20	x	fixed	✕
dKdGdyDom	✔	✔	Texture	8D28	x	fixed	✕
dKdRdx	✔	✔	Texture	8D08	x	fixed	✕
dKdRdyDom	✔	✔	Texture	8D10	x	fixed	✕
dKsBdx	✔	✔	Texture	8CB8	x	fixed	✕
dKsBdyDom	✔	✔	Texture	8CC0	x	fixed	✕
dKsdx	✔	✔	Texture	86D0	x	fixed	✕
dKsdyDom	✔	✔	Texture	86D8	x	fixed	✕
dKsGdx	✔	✔	Texture	8CA0	x	fixed	✕
dKsGdyDom	✔	✔	Texture	8CA8	x	fixed	✕
dKsRdx	✔	✔	Texture	8C88	x	fixed	✕
dKsRdyDom	✔	✔	Texture	8C90	x	fixed	✕
DMAAddr	✕	✔	Host In	A980	x	integer	✕
DMAContinue	✕	✔	Host In	A9F8	x	integer	✔
DMACount	✕	✔	Host In	A988	x	integer	✕
DMAFeedback	✕	✔	Host In	AA10	x	integer	✕
DMAMemoryControl	✔	✔	Host In	B780	x	bitfield	✕
DMAOutput Address	✕	✔	Host In	A9E0	x	integer	✕
DMAOutputCount	✕	✔	Host In	A9E8	x	integer	✕
DMARectangle Read	✕	✔	Host In	A9A8	x	bitfield	✕
DMARectangleRead LinePitch	✔	✔	Host In	A9B8	x	integer	✕
DMARectangleRead Target	✔	✔	Host In	A9C0	x	bitfield	✕
DMARectangleReadAddress	✔	✔	Host In	A9B0	x	integer	✕

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
DMARectangleWrite	✗	✔	Host In	A9C8	x	bitfield	✗
DMARectangleWrite Address	✔	✔	Host In	A9D0	x	integer	✗
DMARectangleWriteLinePitch	✔	✔	Host In	A9D8	x	integer	✗
DownloadGlyphWidth	✔	✔	2D Set Up	B658	x	integer	✗
DownloadTarget	✔	✔	2D Set Up	B650	x		✔
dQ1dx	✔	✔	Texture coord	8438	x	fixed	✗
dQ1dyDom	✔	✔	Texture coord	8440	x	fixed	✗
dQdx	✔	✔	Texture coord	83C0	x	fixed	✗
DQdy	✔	✔	Texture coord	83E8	x	fixed	✗
dQdyDom	✔	✔	Texture coord	83C8	x		✗
DrawLine0	✗	✔	Delta	9318	x	fixed	✔
DrawLine1	✗	✔	Delta	9320	x	fixed	✔
DrawLine2D01	✗	✔	Delta	9778	x	bitfield	✔
DrawLine2D10	✗	✔	Delta	9768	x	bitfield	✔
DrawPoint	✗	✔	Delta	9330	x	bitfield	✔
DrawTriangle	✗	✔	Delta	9308	x	bitfield	✔
dRdx	✔	✔	Color DDA	8788	x	fixed	✗
dRdyDom	✔	✔	Color DDA Delta	8790	x	fixed	✗
dS1dx	✔	✔	Texture coord	8408	x	fixed	✗
dS1dyDom	✔	✔	Texture coord	8410	x	fixed	✗
dSdx	✔	✔	Texture coord	8390	x	fixed	✗
dSdy	✔	✔	Texture coord	83D8	x	fixed	✗
dSdyDom	✔	✔	Texture coord	8398	x	fixed	✗
dT1dx	✔	✔	Texture coord	8420	x	fixed	✗
dT1dyDom	✔	✔	Texture coord	8428	x	fixed	✗
dTdx	✔	✔	Texture coord	83A8	x	fixed	✗
dTdy	✔	✔	Texture coord	83E0	x	fixed	✗
dTdyDom	✔	✔	Texture coord	83B0	x	fixed	✗

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
dXDom	✔	✕	Raster-izer	8008	x	fixed	✕
dXSub	✔	✕	Raster-izer	8018	x	fixed	✕
dY	✔	✕	Raster-izer	8028	x	fixed	✕
dZdxL	✔	✔	Depth & Fog	89C8	x	fixed	✕
dZdxU	✔	✔	Depth & Fog	89C0	x	fixed	✕
dZdyDomL	✔	✔	Depth & Fog	89D8	x	bitfield	✕
dZdyDomU	✔	✔	Depth & Fog	89D0	x	fixed	✕
EndOfFeedback	✔	✔	Host Out	8FF8	x	unused	✕
ErrorFlags			Control Status	0038		bitfield	
FastClearDepth	✔	✔	Depth	89E0	x	integer	✕
FBBlockColor	✔	✕	FB Read	8AC8	x	integer	✕
FBBlockColor[0…3]	✔	✔	FB Write	B060	x	integer	✕
FBBlockColorBack	✔	✔	FB Write	B0A0	x	integer	✕
FBBlockColorBack[0…3]	✔	✔	FB Write	B080	x	integer	✕
FBColor	0	✕	FB Write	8A98	x	n/a	✕
FBDestReadBufferAddr[0…3]	✔	✔	FB Read	AE80	x	integer	✕
FBDestReadBufferOffset[0…3]	✔	✔	FB Read	AEA0	x	integer	✕
FBDestReadBufferWidth[0…3]	✔	✔	FB Read	AEC0	x	integer	✕
FBDestReadEnables	✔	✔	FB Read	AEE8	x	bitfield	✕
FBDestReadEnablesAnd	✕	✔	FB Read	AD20	x	bitfield	✕
FBDestReadEnablesOr	✕	✔	FB Read	AD28	x	bitfield	✕
FBDestReadMode	✔	✔	FB Read	AEE0	x	bitfield	✕
FBDestReadModeAnd	✕	✔	FB Read	AC90	x	bitfield	✕
FBDestReadModeOr	✕	✔	FB Read	AC98	x	bitfield	✕
FBHardwareWriteMask	✔	✔	FB Write	8AC0	x	mask	✕
FBSoftwareWriteMask	✔	✔	Logic Ops	8820	x	integer	✕
FBSourceReadBufferAddr	✔	✔	FB Read	AF08	x	integer	✕
FBSourceReadBufferOffset	✔	✔	FB Read	AF10	x	integer	✕
FBSourceReadBufferWidth	✔	✔	FB Read	AF18	x	integer	✕
FBSourceReadMode	✔	✔	FB Read	AF00	x	bitfield	✕
FBSourceReadModeAnd	✕	✔	FB Read	ACA0	x	bitfield	✕
FBSourceReadModeOr	✕	✔	FB Read	ACA8	x	bitfield	✕
FBWriteBufferAddr[0…3]	✔	✔	FB Write	B000	x	integer	✕
FBWriteBufferOffset[0…3]	✔	✔	FB Write	B020	x	integer	✕
FBWriteBufferWidth[0…3]	✔	✔	FB Write	B040	x	integer	✕
FBWriteMode	✔	✔	FB Write	8AB8	x	bitfield	
FBWriteModeAnd	✕	✔	FB Write	ACF0	x	bitfield	✕

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
FBWriteModeOr	✗	✔	FB Write	ACF8	x	bitfield	✗
FeedbackX		✔	Host Out	8F88	x	integer	✗
FeedbackY		✔	Host Out	8F90	x	integer	✗
FifoControl	✔	✔	Video Control	3078		bitfield	
FIFODiscon	✔	✔	Control Status	0068		bitfield	
FillBackgroundColor	✗	✔	2D Set Up	8330	x	integer	✗
FillConfig2D0	✗	✔	2D Set Up	8338	x	bitfield	✗
FillConfig2D1	✗	✔	2D Set Up	8360	x	bitfield	
FillFBDestReadBufferAddr0	✗	✔	2D Set Up	8310	x	integer	✗
FillFBSourceReadBufferAddr	✗	✔	2D Set Up	8308	x	integer	✗
FillFBSourceReadBufferOffset	✗	✔	2D Set Up	8340	x	integer	✗
FillFBWriteBufferAddr0	✗	✔	2D Set Up	8300	x	integer	✗
FillForegroundColor0	✗	✔	2D Set Up	8328	x	integer	✗
FillForegroundColor1	✗	✔	2D Set Up	8358	x	integer	✗
FillGlyphPosition	✗	✔	2D Set Up	8368	x	integer	✗
FillRectanglePosition	✗	✔	2D Set Up	8348	x	integer	✗
FillRender2D	✗	✔	2D Set Up	8350	x	bitfield	✗
FillScissorMaxXY	✗	✔	2D Set Up	8320	x	fixed	✗
FillScissorMinXY	✗	✔	2D Set Up	8318	x	fixed	✗
FilterMode	✔	✔	Host Out	8C00	x	bitfield	✗
FilterModeAnd	✗	✔	Host Out	AD00	x	bitfield	✗
FilterModeOr	✗	✔	Host Out	AD08	x	bitfield	✗
FlushSpan	✗	✔	Raster-izer	8060	x	tag	✔
FlushWriteCombining	✗	✔	Host In	8910	x	integer	✗
FogColor	✔	✔	Fog	8698	x	fixed	✗
FogMode	✔	✔	Fog	8690	x	bitfield	✗
FogModeAnd	✗	✔	Fog	AC10	x	bitfield	✗
FogModeOr	✗	✔	Fog	AC18	x	bitfield	✗
FogTable[0…15]	✔	✔	Fog	B100	x	bitfield	✗
FogTable[16…31]	✔	✔	Fog	B180	x	bitfield	✗
FogTable[32…47]	✔	✔	Fog	B200	x	bitfield	✗

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
FogTable[48…63]	✔	✔	Fog	B280	x	bitfield	✕
ForegroundColor	✔	✔	Logic Ops	B0C0	x	integer	✕
FStart	✔	✔	Fog	86A0	x	fixed	✕
GIDMode	✔	✔	LB Read	B538	x	bitfield	✕
GIDModeAnd	✕	✔	LB Read	B5B0	x	bitfield	✕
GIDModeOr	✕	✔	LB Read	B5B8	x	bitfield	✕
GlyphData	✕	✔	2D Set Up	B660	x	integer	✕
GlyphPosition	✔	✔	2D Set Up	B608	x	integer	✕
GPOutDMAAddress	✔	✕	Control Status	0080		integer	
GStart	✔	✔	Color DDA	8798	x	fixed	✕
HbEnd	✔	✔	Video Control	3020		integer	
HeadPhysicalPage Allocation[0…3]	✔	✔	Texture Read	B480	x	integer	✕
HgEnd	✔	✔	Video Control	3018		integer	
HostInDMAAddress	✔	✕	Host In	8938	x	integer	✕
HostInID	✔	✔	Host In	8900	x		✕
HostInState	✔	✔	Host In	8918	x	integer	✕
HostInState2	✔	✔	Host In	8940	x	integer	✕
HostTextureAddress	✔	✕	Control Status	0100		integer	
HsEnd	✔	✔	Video Control	3030		integer	
HsStart	✔	✔	Video Control	3028		integer	
HTotal	✔	✔	Video Control	3010		integer	
IndexBaseAddress	✔	✔	Host In	B700	x	integer	✕
IndexedDoubleVertex	✕	✔	Host In	B7B0	x	integer	✕
IndexedLineList	✕	✔	Host In	B728	x	integer	✕
IndexedLineStrip	✕	✔	Host In	B730	x	integer	✕
IndexedPointList	✕	✔	Host In	B738	x	integer	✕
IndexedPolygon	✕	✔	Host In	B740	x	integer	✕
IndexedTriangleFan	✕	✔	Host In	B718	x	integer	✕
IndexedTriangleList	✕	✔	Host In	B710	x	integer	✕
IndexedTriangleStrip	✕	✔	Host In	B720	x	integer	✕
IndexedVertex	✕	✔	Host In	B7A8	x	integer	✕
InFIFOSpace	✔	✕	Control Status	0018		integer	
IntEnable	✔	✔	Control Status	0008		bitfield	

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
InterruptLine	✔	✔	Video Control	3060		integer	
IntFlags	✔	✔	Control Status	0010		bitfield	
InvalidateCache	✕	✔	Texture Read	B358	x	bitfield	✔
KdBStart	✔	✔	Texture	8D30	x	fixed	✕
KdGStart	✔	✔	Texture	8D18	x	fixed	✕
KdRStart	✔	✔	Texture	8D00	x	fixed	✕
KdStart	✔	✔	Texture	86E0	x	fixed	✕
KsBStart	✔	✔	Texture Application	8CB0	x	fixed	✕
KsGStart	✔	✔	Texture Application	8C98	x	fixed	✕
KsRStart	✔	✔	Texture Application	8C80	x	fixed	✕
KsStart	✔	✔	Texture	86C8	x	fixed	✕
LBClearDataL	✔	✔	LB Read	B550	x	integer	✕
LBClearDataU	✔	✔	LB Read	B558	x	integer	✕
LBDepth	✕	✔	LB Read/Host Out	88B0	x	integer	✕
LBDestReadBufferAddr	✔	✔	LB Read	B510	x	integer	
LBDestReadBufferOffset	✔	✔	LB Read	B518	x	integer	
LBDestReadEnables	✔	✔	LB Read	B508	x	bitfield	✕
LBDestReadEnables And	✕	✔	LB Read	B590	x	bitfield	✕
LBDestReadEnables Or	✕	✔	LB Read	B598	x	bitfield	✕
LBDestReadMode	✔	✔	LB Read	B500	x	integer	✕
LBDestReadModeAnd	✕	✔	LB Read	B580	x	bitfield	✕
LBDestReadModeOr	✕	✔	LB Read	B588	x	bitfield	✕
LBReadFormat	✔	✔	LB Read	8888	x	bitfield	✕
LBSourceReadBufferAddr	✔	✔	LB Read	B528	x	integer	✕
LBSourceReadBufferOffset	✔	✔	LB Read	B530	x	bitfield	✕
LBSourceReadMode	✔	✔	LB Read	B520	x	integer	✕
LBSourceReadMode And	✕	✔	LB Read	B5A0	x	bitfield	✕
LBSourceReadModeOr	✕	✔	LB Read	B5A8	x	bitfield	✕
LBStencil	✕	✔	Host Out	88A8	x	bitfield	✕
LBWriteBufferAddr	✔	✔	LB Write	B540	x	integer	✕
LBWriteBufferOffset	✔	✔	LB Write	B548	x	integer	✕
LBWriteFormat	✔	✔	LB Write	88C8	x	bitfield	✕
LBWriteMode	✔	✔	LB Write	88C0	x	bitfield	✕
LBWriteModeAnd	✕	✔	LB Write	AC80	x	bitfield	✕
LBWriteModeOr	✕	✔	LB Write	AC88	x	bitfield	✕

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
LineCoord0	✗	✔	Delta	9760	x	bitfield	✗
LineCoord1	✗	✔	Delta	9770	x	bitfield	✗
LineMode	✔	✔	Delta	94A8	x	bitfield	✗
LineModeAnd	✗	✔	Delta	AAF0	x	bitfield	✗
LineModeOr	✗	✔	Delta	AAF8	x	bitfield	✗
LineStippleMode	✔	✔	Stipple	81A8	x	Bitfield	
LineStippleModeAnd	✗	✔	Stipple	ABC0	x	bitfield	✗
LineStippleModeOr	✗	✔	Stipple	ABC8	x	bitfield	✗
LineWidth	✔	✔	Delta	94B0	x	integer	✗
LineWidthOffset	✔	✔	Delta	94B8	x	integer	✗
LoadLineStippleCounters	✔	✔	Stipple	81B0	x	Bitfield	✔
LocalMemCaps	✔	✔	Memory Control	1018		Bitfield	
LocalMemControl	✔	✔	Memory Control	1028		Bitfield	
LocalMemPowerDown	✔	✔	Memory Control	1038		Bitfield	
LocalMemRefresh	✔	✔	Memory Control	1030		Bitfield	
LocalMemTiming	✔	✔	Memory Control	1020		Bitfield	
LOD	✔	✔	Texture Index	83D0	x	fixed	✗
LOD1	✔	✔	Texture Index	8448	x	fixed	✗
LodRange0	✔	✔	Texture Index	B348	x	bitfield	✗
LodRange1	✔	✔	Texture Index	B350	x	fixed	✗
LogicalOpMode	✔	✔	Logic Op	8828	x	bitfield	✗
LogicalOpModeAnd	✗	✔	Logic Op	ACE0	x	bitfield	✗
LogicalOpModeOr	✗	✔	Logic Op	ACE8	x	bitfield	✗
LogicalTexturePage	✔	✗	Control Status	0118		integer	
LogicalTexturePage TableAddr	✔	✔	Texture Read	B4D0	x	integer	✗
LogicalTexturePage TableLength	✔	✔	Texture Read	B4D8	x	integer	✗
LUT[0…15]	✔	✔	LUT	8E80	x	bitfield	✗
LUTAddress	✔	✔	Texture Read	84D0	x	integer	✗
LUTData	✔	✔	LUT	84C8	x	integer	✗
LUTIndex	✔	✔	LUT	84C0	x	integer	✗
LUTMode	✔	✔	LUT	B378	x	bitfield	✗
LUTModeAnd	✗	✔	LUT	AD70	x	bitfield	✗
LUTModeOr	✗	✔	LUT	AD78	x	bitfield	✗

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
LUTTransfer	✔	✔	Texture Read	84D8	x	bitfield	✕
MaxHitRegion	✕	✔	Host Out	8C30	x	bitfield	✔
MaxRegion	✔	✔	Host Out	8C18	x	bitfield	
MemBypassWriteMask	✔	✔	Memory Control	1008		Integer	
MemCounter	✔	✕	Memory Control	1000		Integer	
MemScratch	✔	✔	Memory Control	1010		Integer	
MinHitRegion	✕	✔	Host Out	8C28	x	bitfield	✔
MinRegion	✔	✔	Host Out	8C10	x	bitfield	✕
MiscControl	✔	✔	Video Control	3088		Bitfield	
OutPutFIFOWords	✔	✕	Control Status	0020		integer	
Packed16Pixels	✕	✔	2D Set Up	B638	x	integer	✔
Packed4Pixels	✕	✔	2D Set Up	B668	x	integer	✔
Packed8Pixels	✕	✔	2D Set Up	B630	x	integer	✔
PCIAbortAddress	✔	✕	Control Status	0098		integer	
PCIAbortStatus	✔	✕	Control Status	0090		bitfield	
PCIFeedbackCount	✔	✕	Control Status	0088		integer	
PCIPLLStatus	✔	✔	Control Status	00F0		bitfield	
PhysicalPageAllocationTableAddr	✔	✔	Texture Read	B4C0	x	integer	✕
PickResult	✕	✔	Host Out	8C38	x	bitfield	✔
PixelSize	✔	✔	Raster-izer	80C0	x	Bitfield	✔
PointSize	✔	✔	Delta	9498	x	integer	✕
PointTable[0…3]	✔	✔	Raster-izer	8080	x	bitfield	✕
ProvokingVertex	✔	✔	Delta	9338	x	bitfield	✔
ProvokingVertexMask	✔	✔	Delta	9358	x	bitfield	✕
Q1Start	✔	✔	Texture Coord	8430	x	fixed	✕
QStart	✔	✔	Texture Coord	83B8	x	fixed	✕
RasterizerMode	✔	✔	Raster-izer	80A0	x	Bitfield	✕
RasterizerModeAnd	✕	✔	Raster-izer	ABA0	x	bitfield	✕

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
RasterizerModeOr	✗	✔	Raster-izer	ABA8	x	bitfield	✗
RDIndexControl	✔	✔	RAMDAC Control	4038		Integer	
RDIndexedData	✔	✔	RAMDAC Control	4030		Integer	
RDIndexHigh	✔	✔	RAMDAC Control	4028		Integer	
RDIndexLow	✔	✔	RAMDAC Control	4020		Integer	
RDPaletteData	✔	✔	RAMDAC Control	4008		Integer	
RDPaletteReadAddress	✔	✔	RAMDAC Control	4018		Integer	
RDPaletteWriteAddress	✔	✔	RAMDAC Control	4000		Integer	
RDPixelMask	✔	✔	RAMDAC Control	4010		Integer	
RectangleHeight	✔	✔	Delta	94E0	x	float	✗
RectanglePosition	✔	✔	2D Set Up	B600	x	integer	✗
RemoteMemControl	✔	✔	Memory Control	1100		Integer	
Render	✗	✔	Raster-izer	8038	x	Bitfield	✔
Render2D	✗	✔	2D Set Up	B640	x	bitfield	✗
Render2DGlyph	✗	✔	2D Set Up	B648	x	bitfield	✗
RenderPatchOffset	✔	✔	2D Set Up	B610	x	bitfield	✗
RepeatLine	✗	✔	Delta	9328	x	tag	✔
RepeatTriangle	✗	✔	Delta	9310	x	tag	✔
ResetPickResult	✗	✔	Host Out	8C20	x	tag	✔
ResetStatus			Control Status	0000		integer	
RetainedRender	✔	✔	Host In	B7A0	x	bitfield	✔
RLCount	✗	✔	2D Set Up	B678	x	integer	✗
RLData	✔	✔	2D Set Up	B670	x	integer	✗
RLEMask	✔	✔	Host Out	8C48	x	bitfield	✗
RouterMode	✔	✔	Router	8840	x	bitfield	✗
RStart	✔	✔	Color DDA	8780	x	fixed	✗
S1Start	✔	✔	Texture Coord	8400	x	fixed	✗
SaveLineStippleCounters	✗	✔	Stipple	81C0	x	tag	✔

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
ScissorMaxXY	✔	✔	Scissor	8190	x	Bitfield	✘
ScissorMinXY	✔	✔	Scissor	8188	x	Bitfield	✘
ScissorMode	✔	✔	Scissor	8180	x	Bitfield	✘
ScissorModeAnd	✘	✔	Scissor	ABB0	x	bitfield	✘
ScissorModeOr	✘	✔	Scissor	ABB8	x	bitfield	✘
ScreenBase	✔	✔	Video Control	3000		integer	
ScreenBaseRight	✔	✔	Video Control	3080		Integer	
ScreenSize	✔	✔	Scissor	8198	x	Bitfield	
ScreenStride	✔	✔	Video Control	3008		Integer	
Security	✘	✔	Host In	8908	x	bitfield	✘
SetLogicalTexturePage	✔	✔	Texture Read	B360	x	bitfield	✘
SizeOfFramebuffer	✔	✔	LB Read, FB Read, FB Write	B0A8	x	integer	✘
SStart	✔	✔	Texture Coord	8388	x	fixed	✘
StartXDom	✔	✘	Raster-izer	8000	x	fixed	✘
StartXSub	✔	✘	Raster-izer	8010	x	fixed	✘
StartY	✘	✘	Raster-izer	8020	x	fixed	✘
StatisticMode	✔	✔	Host Out	8C08	x	bitfield	✘
StatisticModeAnd	✘	✔	Host Out	AD10	x	bitfield	✘
StatisticModeOr	✘	✔	Host Out	AD18	x	bitfield	✘
Stencil	✔	✔	Stencil	8998	x	bitfield	✔✘
StencilData	✔	✔	Stencil	8990	x	bitfield	
StencilDataAnd	✘	✔	Stencil	B3E0	x	bitfield	✘
StencilDataOr	✘	✔	Stencil	B3E8	x	bitfield	✘
StencilMode	✔	✔	Stencil	8988	x	bitfield	✘
StencilModeAnd	✘	✔	Stencil	AC60	x	bitfield	✘
StencilModeOr	✘	✔	Stencil	AC68	x	bitfield	✘
StripeOffsetY	✔	✔	Raster-izer	80C8	x	fixed	✘
SuspendUntilFrameBlank	✘	✔	Framebuffer Write	8C78	x	bitfield	✔
Sync	✘	✔	Host Out	8C40	x	bitfield	✔
T1Start	✔	✔	Texture coord	8418	x	fixed	✘
TailPhysicalPage Allocation[0…3]	✔	✔	Texture Read	B4A0	x	integer	✘
TexDMAAddress	✔	✘	Control Status	0120		integer	

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
TexFIFOSpace	✔	✕	Control Status	0128		integer	
TextRender2DGlyph0	✕	✔	Raster-izer	8708	x	bitfield	✔
TextRender2DGlyph1	✕	✔	Raster-izer	8718	x	bitfield	✔
TextRender2DGlyph2	✕	✔	Raster-izer	8728	x	bitfield	✔
TextRender2DGlyph3	✕	✔	Raster-izer	8738	x	bitfield	✔
TextRender2DGlyph4	✕	✔	Raster-izer	8748	x	bitfield	✔
TextRender2DGlyph5	✕	✔	Raster-izer	8758	x	bitfield	✔
TextRender2DGlyph6	✕	✔	Raster-izer	8768	x	bitfield	✔
TextRender2DGlyph7	✕	✔	Raster-izer	8778	x	bitfield	✔
TextTGlyphAddr0	✕	✔	Raster-izer	8700	x	integer	✕
TextTGlyphAddr1	✕	✔	Raster-izer	8710	x	integer	✕
TextTGlyphAddr2	✕	✔	Raster-izer	8720	x	integer	✕
TextTGlyphAddr3	✕	✔	Raster-izer	8730	x	integer	✕
TextTGlyphAddr4	✕	✔	Raster-izer	8740	x	integer	✕
TextTGlyphAddr5	✕	✔	Raster-izer	8750	x	integer	✕
TextTGlyphAddr6	✕	✔	Raster-izer	8760	x	integer	✕
TextTGlyphAddr7	✕	✔	Raster-izer	8770	x	integer	✕
TextureApplication ModeAnd	✕	✔	Texture Application	AC50	x	bitfield	✕
TextureApplication ModeOr	✕	✔	Texture Application	AC58	x	bitfield	✕
TextureApplicationMode	✔	✔	Texture Application	8680	x	bitfield	✕
TextureBaseAddr[16]	✔	✔	Texture Read	8500	x	integer	✕
TextureCacheControl	✔	✔	Texture	8490	x	bitfield	✕
TextureChromaLower0	✔	✔	Color DDA	84F0	x	bitfield	✕

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
TextureChromaLower1	✔	✔	Texture Filter	8608	x	bitfield	✕
TextureChromaUpper0	✔	✔	Color DDA	84E8	x	bitfield	✕
TextureChromaUpper1	✔	✔	Texture Filter	8600	x	bitfield	✕
TextureCompositeAlphaMode0	✔	✔	Texture Composite	B310	x	bitfield	✕
TextureCompositeAlphaMode0And	✕	✔	Texture Composite	B390	x	bitfield	✕
TextureCompositeAlphaMode0Or	✕	✔	Texture Composite	B398	x	bitfield	✕
TextureCompositeAlphaMode1	✔	✔	Texture Composite	B320	x		✕
TextureCompositeAlphaMode1And	✕	✔	Texture Composite	B3B0	x	bitfield	✕
TextureCompositeAlphaMode1Or	✕	✔	Texture Composite	B3B8	x	bitfield	✕
TextureCompositeColorMode0	✔	✔	Texture Composite	B308	x	bitfield	✕
TextureCompositeColorMode0And	✕	✔	Texture Composite	B380	x	bitfield	✕
TextureCompositeColorMode0Or	✕	✔	Texture Composite	B388	x	bitfield	✕
TextureCompositeColorMode1	✔	✔	Texture Composite	B318	x	bitfield	✕
TextureCompositeColorMode1And	✕	✔	Texture Composite	B3A0	x	bitfield	✕
TextureCompositeColorMode1Or	✕	✔	Texture Composite	B3A8	x	bitfield	✕
TextureCompositeFactor0	✔	✔	Texture Composite	B328	x	bitfield	
TextureCompositeFactor1	✔	✔	Texture Composite	B330	x	bitfield	✕

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
TextureCompositeMode	✔	✔	Texture Composite	B300	x	bitfield	✕
TextureCoordMode	✔	✔	Texture coord	8380	x	bitfield	✕
TextureCoordModeAnd	✕	✔	Texture coord	AC20	x		✕
TextureCoordModeOr	✕	✔	Texture coord	AC28	x	bitfield	✕
TextureData	✕	✔	Localbuffer R/W	88E8	x	integer	✕
TextureDownloadControl	✔	x	Control Status	0108		bitfield	
TextureDownloadOffset	✔	✔	Localbuffer R/W	88F0	x		✕
TextureEnvColor	✔	✔	Texture	8688	x	bitfield	✕
TextureFilterMode	✔	✔	Texture	84E0	x	bitfield	✕
TextureFilterModeAnd	✕	✔	Texture	AD50	x	bitfield	✕
TextureFilterModeOr	✕	✔	Texture	AD58	x	bitfield	✕
TextureIndexMode0	✔	✔	Texture Index	B338	x	bitfield	✕
TextureIndexMode0And	✕	✔	Texture Index	B3C0	x	bitfield	✕
TextureIndexMode0Or	✕	✔	Texture Index	B3C8	x	bitfield	✕
TextureIndexMode1	✔	✔	Texture Index	B340	x	bitfield	✕
TextureIndexMode1And	✕	✔	Texture Index	B3D0	x	bitfield	✕
TextureIndexMode1Or	✕	✔	Texture Index	B3D8	x	bitfield	✕
TextureLodBiasS	✔	✔	Texture Index	8450	x	fixed	✕
TextureLodBiasT	✔	✔	Texture Index	8458	x	fixed	✕
TextureLODScale	✔	✔	Texture coord	9340	x	float	✕
TextureLODScale1	✔	✔	Texture coord	9348	x	float	✕
TextureMapSize	✔	✔	Texture Read	B428	x	integer	✕
TextureMapWidth[16]	✔	✔	Texture Read	8580	x	bitfield	✕
TextureOperation	✔	✕	Control Status	0110		integer	
TextureReadMode0	✔	✔	Texture Read	B400	x	bitfield	✕
TextureReadMode0And	✕	✔	Texture Read	AC30	x	bitfield	✕

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
TextureReadMode0Or	✕	✔	Texture Read	AC38	x	bitfield	✕
TextureReadMode1	✔	✔	Texture Read	B408	x	bitfield	✕
TextureReadMode1And	✕	✔	Texture Read	AD40	x	bitfield	✕
TextureReadMode1Or	✕	✔	Texture Read	AD48	x	bitfield	✕
TouchLogicalPage	✕	✔	Texture Read	B370	x	bitfield	✔
TStart	✔	✔	Texture coord	83A0	x	fixed	✕
UpdateLineStippleCounters	✕	✔	Stipple	81B8	x	Bitfield	✔
UpdateLogicalTextureInfo	✕	✔	Texture Read	B368	x	tag	✔
V0FloatA	✔	✔	Delta	91C0	x	float	✕
V0FloatB	✔	✔	Delta	91B8	x	float	✕
V0FloatF	✔	✔	Delta	91C8	x	float	✕
V0FloatG	✔	✔	Delta	91B0	x	float	✕
V0FloatKdB	✔	✔	Delta	9078	x	float	✕
V0FloatKdG	✔	✔	Delta	9070	x	float	✕
V0FloatKdR	✔	✔	Delta	9068	x	float	✕
V0FloatKsB	✔	✔	Delta	9060	x	float	
V0FloatKsG	✔	✔	Delta	9058	x	float	✕
V0FloatKsR	✔	✔	Delta	9050	x	float	✕
V0FloatPackedColor	✕	✔	Delta	91F0	x	bitfield	✕
V0FloatPackedDiffuse	✕	✔	Delta	9048	x	bitfield	✕
V0FloatPackedSpecularFog	✕	✔	Delta	91F8	x	bitfield	✕
V0FloatQ	✔	✔	Delta	9190	x	float	✕
V0FloatQ1	✔	✔	Delta	9010	x	float	✕
V0FloatR	✔	✔	Delta	91A8	x	float	✕
V0FloatS	✔	✔	Delta	9180	x	float	✕
V0FloatS1	✔	✔	Delta	9000	x	float	✕
V0FloatT	✔	✔	Delta	9188	x	float	✕
V0FloatT1	✔	✔	Delta	9008	x	float	✕
V0FloatX	✔	✔	Delta	91D0	x	float	✕
V0FloatY	✔	✔	Delta	91D8	x	float	✕
V0FloatZ	✔	✔	Delta	91E0	x	float	✕
V1FloatA	✔	✔	Delta	9240	x	float	✕
V1FloatB	✔	✔	Delta	9238	x	float	✕
V1FloatF	✔	✔	Delta	9248	x	float	✕
V1FloatG	✔	✔	Delta	9230	x	float	✕
V1FloatKdB	✔	✔	Delta	90F8	x	float	✕
V1FloatKdG	✔	✔	Delta	90F0	x	float	✕
V1FloatKdR	✔	✔	Delta	90E8	x	float	✕
V1FloatKsB	✔	✔	Delta	90E0	x	float	✕

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
V1FloatKsG	✔	✔	Delta	90D8	x	float	✕
V1FloatKsR	✔	✔	Delta	90D0	x	float	✕
V1FloatPackedColor	✕	✔	Delta	9270	x	bitfield	✕
V1FloatPackedDiffuse	✕	✔	Delta	90C8	x	bitfield	✕
V1FloatPackedSpecularFog	✕	✔	Delta	9278	x	bitfield	✕
V1FloatQ	✔	✔	Delta	9210	x	float	✕
V1FloatQ1	✔	✔	Delta	9090	x	float	✕
V1FloatR	✔	✔	Delta	9228	x	float	✕
V1FloatS	✔	✔	Delta	9200	x	float	✕
V1FloatS1	✔	✔	Delta	9080	x	float	✕
V1FloatT	✔	✔	Delta	9208	x	float	✕
V1FloatT1	✔	✔	Delta	9088	x	float	✕
V1FloatX	✔	✔	Delta	9250	x	float	✕
V1FloatY	✔	✔	Delta	9258	x	float	✕
V1FloatZ	✔	✔	Delta	9260	x	float	✕
V2FloatA	✔	✔	Delta	92C0	x	float	✕
V2FloatB	✔	✔	Delta	92B8	x	float	✕
V2FloatF	✔	✔	Delta	92C8	x	float	✕
V2FloatG	✔	✔	Delta	92B0	x	float	✕
V2FloatKdB	✔	✔	Delta	9178	x	float	✕
V2FloatKdG	✔	✔	Delta	9170	x	float	✕
V2FloatKdR	✔	✔	Delta	9168	x	float	✕
V2FloatKsB	✔	✔	Delta	9160	x	float	✕
V2FloatKsG	✔	✔	Delta	9158	x	float	✕
V2FloatKsR	✔	✔	Delta	9150	x	float	✕
V2FloatPackedColor	✕	✔	Delta	92F0	x	bitfield	✕
V2FloatPackedDiffuse	✕	✔	Delta	9148	x	bitfield	✕
V2FloatPackedSpecularFog	✕	✔	Delta	92F8	x	bitfield	✕
V2FloatQ	✔	✔	Delta	9290	x	float	✕
V2FloatQ1	✔	✔	Delta	9110	x	float	✕
V2FloatR	✔	✔	Delta	92A8	x	float	✕
V2FloatS	✔	✔	Delta	9280	x	float	✕
V2FloatS1	✔	✔	Delta	9100	x	float	✕
V2FloatT	✔	✔	Delta	9288	x	float	✕
V2FloatT1	✔	✔	Delta	9108	x	float	✕
V2FloatX	✔	✔	Delta	92D0	x	float	✕
V2FloatY	✔	✔	Delta	92D8	x	float	✕
V2FloatZ	✔	✔	Delta	92E0	x	float	✕
VbEnd	✔	✔	Video Control	3040	x	integer	
VClkRDacCtl	✔	✔	Control Status	0040	0	bitfield	
Vertex0	✕	✔	Host In	B7B8	x	integer	✕
Vertex1	✕	✔	Host In	B7C0	x	integer	✕
Vertex2	✕	✔	Host In	B7C8	x	integer	✕

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
VertexBaseAddress	✔	✔	Host In	B708	x	integer	✕
VertexControl	✔	✔	Host In	B798	x	bitfield	✕
VertexData	✕	✔	Host In	B7E8	x	integer	✕
VertexData0	✕	✔	Host In	B7D0	x	integer	✕
VertexData1	✕	✔	Host In	B7D8	x	integer	✕
VertexData2	✕	✔	Host In	B7E0	x	integer	✕
VertexFormat	✔	✔	Host In	B790	x	integer	✕
VertexLineList	✕	✔	Host In	B760	x	integer	✕
VertexLineStrip	✕	✔	Host In	B768	x	integer	✕
VertexPointList	✕	✔	Host In	B770	x	integer	✕
VertexPolygon	✕	✔	Host In	B778	x	integer	✕
VertexTagList[0…15]	✔	✔	Host In	B800	x	bitfield	✕
VertexTagList[16…31]	✔	✔	Host In	B880	x	bitfield	✕
VertexTriangleFan	✕	✔	Host In	B750	x	integer	✕
VertexTriangleList	✕	✔	Host In	B748	x	integer	✕
VertexTriangleStrip	✕	✔	Host In	B758	x	integer	✕
VertexValid	✔	✔	Host In	B788	x	integer	✕
VerticalLineCount	✔	✕	Video Control	3070		integer	
VideoControl	✔	✔	Video Control	3058		bitfield	
VideoOverlayBase0	✔	✔	Video Overlay Control	3120		bitfield	
VideoOverlayBase1	✔	✔	Video Overlay Control	3128		bitfield	
VideoOverlayBase2	✔	✔	Video Overlay Control	3130		bitfield	
VideoOverlayFieldOffset	✔	✔	Video Overlay Control	3170		bitfield	
VideoOverlayFIFO Control	✔	✔	Video Overlay Control	3110		bitfield	
VideoOverlayHeight	✔	✔	Video Overlay Control	3148		integer	
VideoOverlayIndex	✔	✔	Video Overlay Control	3118		bitfield	
VideoOverlayMode	✔	✔	Video Overlay Control	3108		bitfield	
VideoOverlayOrigin	✔	✔	Video Overlay Control	3150		bitfield	

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
VideoOverlayShrinkXDelta	✔	✔	Video Overlay Control	3158		bitfield	
VideoOverlayStatus	✔	✔	Video Overlay Control	3178		bitfield	
VideoOverlayStride	✔	✔	Video Overlay Control	3138		integer	
VideoOverlayUpdate	✔	✔	Video Overlay Control	3100		integer	
VideoOverlayWidth	✔	✔	Video Overlay Control	3140		integer	
VideoOverlayYDelta	✔	✔	Video Overlay Control	3168		Integer	
VideoOverlayZoomXDelta	✔	✔	Video Overlay Control	3160		integer	
VSAControl	✔	✔	Video Stream Control	5900		bitfield	
VSACurrentLine	✔	✕	Video Stream Control	5910		integer	
VSAFifoControl	✔	✔	Video Stream Control	59B8		bitfield	
VSAInterruptLine	✔	✔	Video Stream Control	5908		Integer	
VSATimeStamp0	✔	✕	Video Stream Control	59C0		integer	
VSATimeStamp1	✔	✕	Video Stream Control	59C8		integer	
VSATimeStamp2	✔	✕	Video Stream Control	59D0		integer	
VSAVBIAddress0	✔	✔	Video Stream Control	5978		integer	
VSAVBIAddress1	✔	✔	Video Stream Control	5980		integer	

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
VSAVBIAddress2	✔	✔	Video Stream Control	5988		integer	
VSAVBIAddressHost	✔	✔	Video Stream Control	5968		integer	
VSAVBIAddressIndex	✔	✕	Video Stream Control	5970		integer	
VSAVBIEndData	✔	✔	Video Stream Control	59B0		integer	
VSAVBIEndLine	✔	✔	Video Stream Control	59A0		integer	
VSAVBIStartData	✔	✔	Video Stream Control	59A8		integer	
VSAVBIStartLine	✔	✔	Video Stream Control	5998		integer	
VSAVBIStride	✔	✔	Video Stream Control	5990		integer	
VSAVideoAddress0	✔	✔	Video Stream Control	5928		integer	
VSAVideoAddress1	✔	✔	Video Stream Control	5930		integer	
VSAVideoAddress2	✔	✔	Video Stream Control	5938		integer	
VSAVideoAddressHost	✔	✔	Video Stream Control	5918		integer	
VSAVideoAddressIndex	✔	✔	Video Stream Control	5920		integer	
VSAVideoEndData	✔	✔	Video Stream Control	5960		integer	
VSAVideoEndLine	✔	✔	Video Stream Control	5950		integer	
VSAVideoStartData	✔	✔	Video Stream Control	5958		integer	

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
VSAVideoStartLine	✔	✔	Video Stream Control	5948		integer	
VSAVideoStride	✔	✔	Video Stream Control	5940		integer	
VSBControl	✔	✔	Video Stream Control	5A00		bitfield	
VSBCurrentLine	✔	✔	Video Stream Control	5A10		integer	
VSBFifoControl	✔	✔	Video Stream Control	5AB8		bitfield	
VSBInterruptLine	✔	✔	Video Stream Control	5A08		integer	
VSBVBIAddress0	✔	✔	Video Stream Control	5A78		integer	
VSBVBIAddress1	✔	✔	Video Stream Control	5A80		integer	
VSBVBIAddress2	✔	✔	Video Stream Control	5A88		integer	
VSBVBIAddressHost	✔	✔	Video Stream Control	5A68		integer	
VSBVBIAddressIndex	✔	✕	Video Stream Control	5A70	0x2	integer	
VSBVBIEndData	✔	✔	Video Stream Control	5AB0		integer	
VSBVBIEndLine	✔	✔	Video Stream Control	5AA0		integer	
VSBVBIStartData	✔	✔	Video Stream Control	5AA8		integer	
VSBVBIStartLine	✔	✔	Video Stream Control	5A98		integer	
VSBVBIStride	✔	✔	Video Stream Control	5A90		integer	

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
VSBVideoAddress0	✔	✔	Video Stream Control	5A28		integer	
VSBVideoAddress1	✔	✔	Video Stream Control	5A30		integer	
VSBVideoAddress2	✔	✔	Video Stream Control	5A38		integer	
VSBVideoAddressHost	✔	✔	Video Stream Control	5A18		integer	
VSBVideoAddressIndex	✔	✕	Video Stream Control	5A20		integer	
VSBVideoEndData	✔	✔	Video Stream Control	5A60		integer	
VSBVideoEndLine	✔	✔	Video Stream Control	5A50		integer	
VSBVideoStartData	✔	✔	Video Stream Control	5A58		integer	
VSBVideoStartLine	✔	✔	Video Stream Control	5A48		integer	
VSBVideoStride	✔	✔	Video Stream Control	5A40		integer	
VSConfiguration	✔	✔	Video Stream Control	5800		bitfield	
VSDMACommandBase	✔	✔	Video Stream Control	5AC8		integer	
VSDMACommandCount	✔	✔	Video Stream Control	5AD0		integer	
VSDMAMode	✔	✔	Video Stream Control	5AC0		bitfield	
VsEnd	✔	✔	Video Control	3050		integer	
VSSerialBusControl	✔	x	Video Stream Control	5810		bitfield	
VsStart	✔	✔	Video Control	3048		integer	

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
VSStatus	✔	✕	Video Stream Control	5808		bitfield	
VTGAddress	✔	✔	FB Write	B0B0	x	integer	✔
VTGData	✔	✔	FB Write	B0B8	x	integer	✔
VTotal	✔	✔	Video Control	3038		integer	
WaitForCompletion	✕	✔	Raster-izer	80B8	x	Bitfield	✔
Window	✔	✔	Stencil	8980	x	bitfield	✕
WindowAnd	✕	✔	Stencil	AB80	x	bitfield	✕
WindowOr	✕	✔	Stencil	AB88	x	bitfield	✕
WindowOrigin	✔	✔	Scissor	81C8	x	Bitfield	✕
XBias	✔	✔	Delta	9480	x	float	✕
YBias	✔	✔	Delta	9488	x	float	✕
YLimits	✔	✔	Raster-izer	80A8	x	Bitfield	✕
YUVMode	✔	✔	YUV Unit	8F00	x	bitfield	✕
ZFogBias	✔	✔	Fog	86B8	x	float	✕
Zstart	✔	✔	Fog	ADD8	x	integer	✕
ZStartL	✔	✔	Depth & Fog	89B8	x	fixed	✕
ZStartU	✔	✔	Depth	89B0	x	fixed	✕

6.2 Registers Sorted by Offset

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
ResetStatus			Control Status	0000		integer	
IntEnable	✔	✔	Control Status	0008		bitfield	
IntFlags	✔	✔	Control Status	0010		bitfield	
InFIFOSpace	✔	✕	Control Status	0018		integer	
OutPutFIFOWords	✔	✕	Control Status	0020		integer	
ControlDMAAddress	✔	✔	Control Status	0028		integer	
ControlDMACount	✔	✔	Control Status	0030		integer	
ErrorFlags			Control Status	0038		bitfield	
VClkRDacCtl	✔	✔	Control Status	0040	0	bitfield	
ApertureOne	✔	✔	Control Status	0050		bitfield	
ApertureTwo	✔	✔	Control Status	0058		bitfield	
ControlDMAControl	✔	✔	Control Status	0060		bitfield	
FIFODiscon	✔	✔	Control Status	0068		bitfield	
ChipConfig	✔	✔	Control Status	0070		bitfield	
AGPControl	✔	✔	Control Status	0078		bitfield	
GPOutDMAAddress	✔	✕	Control Status	0080		integer	
PCIFeedbackCount	✔	✕	Control Status	0088		integer	
PCIAbortStatus	✔	✕	Control Status	0090		bitfield	
PCIAbortAddress	✔	✕	Control Status	0098		integer	
PCIPLLStatus	✔	✔	Control Status	00F0		bitfield	
HostTextureAddress	✔	✕	Control Status	0100		integer	
TextureDownloadControl	✔	x	Control Status	0108		bitfield	
TextureOperation	✔	✕	Control Status	0110		integer	

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
LogicalTexturePage	✔	✕	Control Status	0118		integer	
TexDMAAddress	✔	✕	Control Status	0120		integer	
TexFIFOSpace	✔	✕	Control Status	0128		integer	
ByAperture1Mode	✔	✔	Bypass Control	0300		Bitfield	
ByAperture1Stride	✔	✔	Bypass Control	0308		Integer	
ByAperture1YStart	✔	✔	Bypass Control	0310		Integer	
ByAperture1UStart	✔	✔	Bypass Control	0318		Integer	
ByAperture1VStart	✔	✔	Bypass Control	0320		Integer	
ByAperture2Mode	✔	✔	Bypass Control	0328		Bitfield	
ByAperture2Stride	✔	✔	Bypass Control	0330		Integer	
ByAperture2YStart	✔	✔	Bypass Control	0338		Integer	
ByAperture2UStart	✔	✔	Bypass Control	0340		Integer	
ByAperture2VStart	✔	✔	Bypass Control	0348		Integer	
ByDMAReadMode	✔	✔	Bypass Control	0350		Bitfield	
ByDMAReadStride	✔	✔	Bypass Control	0358		Integer	
ByDMAReadYStart	✔	✔	Bypass Control	0360		Integer	
ByDMAReadUStart	✔	✔	Bypass Control	0368		Integer	
ByDMAReadVStart	✔	✔	Bypass Control	0370		Integer	
ByDMAReadCommandBase	✔	✔	Bypass Control	0378		Integer	
ByDMAReadCommandCount	✔	✔	Bypass Control	0380		Integer	
ByDMAWriteMode	✔	✔	Bypass Control	0388		Bitfield	
ByDMAWriteStride	✔	✔	Bypass Control	0390		Integer	
ByDMAWriteYStart	✔	✔	Bypass Control	0398		Integer	
ByDMAWriteUStart	✔	✔	Bypass Control	03A0		Integer	
ByDMAWriteVStart	✔	✔	Bypass Control	03A8		Integer	

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
ByDMAWriteCommandBase	✔	✔	Bypass Control	03B0		Integer	
ByDMAWriteCommandCount	✔	✔	Bypass Control	03B8		Integer	
MemCounter	✔	✕	Memory Control	1000		Integer	
MemBypassWriteMask	✔	✔	Memory Control	1008		Integer	
MemScratch	✔	✔	Memory Control	1010		Integer	
LocalMemCaps	✔	✔	Memory Control	1018		Bitfield	
LocalMemTiming	✔	✔	Memory Control	1020		Bitfield	
LocalMemControl	✔	✔	Memory Control	1028		Bitfield	
LocalMemRefresh	✔	✔	Memory Control	1030		Bitfield	
LocalMemPowerDown	✔	✔	Memory Control	1038		Bitfield	
RemoteMemControl	✔	✔	Memory Control	1100		Integer	
ScreenBase	✔	✔	Video Control	3000		integer	
ScreenStride	✔	✔	Video Control	3008		Integer	
HTotal	✔	✔	Video Control	3010		integer	
HgEnd	✔	✔	Video Control	3018		integer	
HbEnd	✔	✔	Video Control	3020		integer	
HsStart	✔	✔	Video Control	3028		integer	
HsEnd	✔	✔	Video Control	3030		integer	
VTotal	✔	✔	Video Control	3038		integer	
VbEnd	✔	✔	Video Control	3040	x	integer	
VsStart	✔	✔	Video Control	3048		integer	
VsEnd	✔	✔	Video Control	3050		integer	
VideoControl	✔	✔	Video Control	3058		bitfield	
InterruptLine	✔	✔	Video Control	3060		integer	
DisplayData			Video Control	3068		bitfield	

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
VerticalLineCount	✔	✕	Video Control	3070		integer	
FifoControl	✔	✔	Video Control	3078		bitfield	
ScreenBaseRight	✔	✔	Video Control	3080		Integer	
MiscControl	✔	✔	Video Control	3088		Bitfield	
VideoOverlayUpdate	✔	✔	Video Overlay Control	3100		integer	
VideoOverlayMode	✔	✔	Video Overlay Control	3108		bitfield	
VideoOverlayFIFO Control	✔	✔	Video Overlay Control	3110		bitfield	
VideoOverlayIndex	✔	✔	Video Overlay Control	3118		bitfield	
VideoOverlayBase0	✔	✔	Video Overlay Control	3120		bitfield	
VideoOverlayBase1	✔	✔	Video Overlay Control	3128		bitfield	
VideoOverlayBase2	✔	✔	Video Overlay Control	3130		bitfield	
VideoOverlayStride	✔	✔	Video Overlay Control	3138		integer	
VideoOverlayWidth	✔	✔	Video Overlay Control	3140		integer	
VideoOverlayHeight	✔	✔	Video Overlay Control	3148		integer	
VideoOverlayOrigin	✔	✔	Video Overlay Control	3150		bitfield	
VideoOverlayShrinkXDelta	✔	✔	Video Overlay Control	3158		bitfield	
VideoOverlayZoomXDelta	✔	✔	Video Overlay Control	3160		integer	
VideoOverlayYDelta	✔	✔	Video Overlay Control	3168		Integer	

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
VideoOverlayFieldOffset	✔	✔	Video Overlay Control	3170		bitfield	
VideoOverlayStatus	✔	✔	Video Overlay Control	3178		bitfield	
RDPaletteWriteAddress	✔	✔	RAMDAC Control	4000		Integer	
RDPaletteData	✔	✔	RAMDAC Control	4008		Integer	
RDPixelMask	✔	✔	RAMDAC Control	4010		Integer	
RDPaletteReadAddress	✔	✔	RAMDAC Control	4018		Integer	
RDIndexLow	✔	✔	RAMDAC Control	4020		Integer	
RDIndexHigh	✔	✔	RAMDAC Control	4028		Integer	
RDIndexedData	✔	✔	RAMDAC Control	4030		Integer	
RDIndexControl	✔	✔	RAMDAC Control	4038		Integer	
VSConfiguration	✔	✔	Video Stream Control	5800		bitfield	
VSStatus	✔	✕	Video Stream Control	5808		bitfield	
VSSerialBusControl	✔	x	Video Stream Control	5810		bitfield	
VSAControl	✔	✔	Video Stream Control	5900		bitfield	
VSAInterruptLine	✔	✔	Video Stream Control	5908		Integer	
VSACurrentLine	✔	✕	Video Stream Control	5910		integer	
VSAVideoAddressHost	✔	✔	Video Stream Control	5918		integer	
VSAVideoAddressIndex	✔	✔	Video Stream Control	5920		integer	
VSAVideoAddress0	✔	✔	Video Stream Control	5928		integer	

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
VSAVideoAddress1	✔	✔	Video Stream Control	5930		integer	
VSAVideoAddress2	✔	✔	Video Stream Control	5938		integer	
VSAVideoStride	✔	✔	Video Stream Control	5940		integer	
VSAVideoStartLine	✔	✔	Video Stream Control	5948		integer	
VSAVideoEndLine	✔	✔	Video Stream Control	5950		integer	
VSAVideoStartData	✔	✔	Video Stream Control	5958		integer	
VSAVideoEndData	✔	✔	Video Stream Control	5960		integer	
VSAVBIAddressHost	✔	✔	Video Stream Control	5968		integer	
VSAVBIAddressIndex	✔	✕	Video Stream Control	5970		integer	
VSAVBIAddress0	✔	✔	Video Stream Control	5978		integer	
VSAVBIAddress1	✔	✔	Video Stream Control	5980		integer	
VSAVBIAddress2	✔	✔	Video Stream Control	5988		integer	
VSAVBIStride	✔	✔	Video Stream Control	5990		integer	
VSAVBIStartLine	✔	✔	Video Stream Control	5998		integer	
VSAVBIEndLine	✔	✔	Video Stream Control	59A0		integer	
VSAVBIStartData	✔	✔	Video Stream Control	59A8		integer	

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
VSAVBIEndData	✔	✔	Video Stream Control	59B0		integer	
VSAFifoControl	✔	✔	Video Stream Control	59B8		bitfield	
VSATimeStamp0	✔	✕	Video Stream Control	59C0		integer	
VSATimeStamp1	✔	✕	Video Stream Control	59C8		integer	
VSATimeStamp2	✔	✕	Video Stream Control	59D0		integer	
VSBControl	✔	✔	Video Stream Control	5A00		bitfield	
VSBInterruptLine	✔	✔	Video Stream Control	5A08		integer	
VSBCurrentLine	✔	✔	Video Stream Control	5A10		integer	
VSBVideoAddressHost	✔	✔	Video Stream Control	5A18		integer	
VSBVideoAddressIndex	✔	✕	Video Stream Control	5A20		integer	
VSBVideoAddress0	✔	✔	Video Stream Control	5A28		integer	
VSBVideoAddress1	✔	✔	Video Stream Control	5A30		integer	
VSBVideoAddress2	✔	✔	Video Stream Control	5A38		integer	
VSBVideoStride	✔	✔	Video Stream Control	5A40		integer	
VSBVideoStartLine	✔	✔	Video Stream Control	5A48		integer	
VSBVideoEndLine	✔	✔	Video Stream Control	5A50		integer	

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
VSBVideoStartData	✔	✔	Video Stream Control	5A58		integer	
VSBVideoEndData	✔	✔	Video Stream Control	5A60		integer	
VSBVBIAddressHost	✔	✔	Video Stream Control	5A68		integer	
VSBVBIAddressIndex	✔	✗	Video Stream Control	5A70	0x2	integer	
VSBVBIAddress0	✔	✔	Video Stream Control	5A78		integer	
VSBVBIAddress1	✔	✔	Video Stream Control	5A80		integer	
VSBVBIAddress2	✔	✔	Video Stream Control	5A88		integer	
VSBVBIStride	✔	✔	Video Stream Control	5A90		integer	
VSBVBIStartLine	✔	✔	Video Stream Control	5A98		integer	
VSBVBIEndLine	✔	✔	Video Stream Control	5AA0		integer	
VSBVBIStartData	✔	✔	Video Stream Control	5AA8		integer	
VSBVBIEndData	✔	✔	Video Stream Control	5AB0		integer	
VSBFifoControl	✔	✔	Video Stream Control	5AB8		bitfield	
VSDMAMode	✔	✔	Video Stream Control	5AC0		bitfield	
VSDMACommandBase	✔	✔	Video Stream Control	5AC8		integer	
VSDMACommandCount	✔	✔	Video Stream Control	5AD0		integer	
StartXDom	✔	✗	Raster-izer	8000	x	fixed	✗

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
dXDom	✔	✕	Raster-izer	8008	x	fixed	✕
StartXSub	✔	✕	Raster-izer	8010	x	fixed	✕
dXSub	✔	✕	Raster-izer	8018	x	fixed	✕
StartY	✕	✕	Raster-izer	8020	x	fixed	✕
dY	✔	✕	Raster-izer	8028	x	fixed	✕
Count	✔	✕	Raster-izer	8030	x	Integer	✕
Render	✕	✔	Raster-izer	8038	x	Bitfield	✔
ContinueNewLine	✕	✔	Raster-izer	8040	x	Integer	✔
ContinueNewDom	✕	✔	Raster-izer	8048	x	Integer	✔
ContinueNewSub	✕	✔	Raster-izer	8050	x	Integer	✔
Continue	✕	✔	Raster-izer	8058	x	Integer	✔
FlushSpan	✕	✔	Raster-izer	8060	x	tag	✔
BitMaskPattern	✕	✔	Raster-izer	8068	x	Integer	✔✕
PointTable[0…3]	✔	✔	Raster-izer	8080	x	bitfield	✕
RasterizerMode	✔	✔	Raster-izer	80A0	x	Bitfield	✕
YLimits	✔	✔	Raster-izer	80A8	x	Bitfield	✕
WaitForCompletion	✕	✔	Raster-izer	80B8	x	Bitfield	✔
PixelSize	✔	✔	Raster-izer	80C0	x	Bitfield	✔
StripeOffsetY	✔	✔	Raster-izer	80C8	x	fixed	✕
ScissorMode	✔	✔	Scissor	8180	x	Bitfield	✕
ScissorMinXY	✔	✔	Scissor	8188	x	Bitfield	✕
ScissorMaxXY	✔	✔	Scissor	8190	x	Bitfield	✕
ScreenSize	✔	✔	Scissor	8198	x	Bitfield	
AreaStippleMode	✔	✔	Stipple	81A0	x	Bitfield	✕
LineStippleMode	✔	✔	Stipple	81A8	x	Bitfield	
LoadLineStippleCounters	✔	✔	Stipple	81B0	x	Bitfield	✔
UpdateLineStippleCounters	✕	✔	Stipple	81B8	x	Bitfield	✔
SaveLineStippleCounters	✕	✔	Stipple	81C0	x	tag	✔
WindowOrigin	✔	✔	Scissor	81C8	x	Bitfield	✕

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
AreaStipplePattern[0…15]	✔	✔	Stipple	8200	x	Bitfield	✕
AreaStipplePattern[16…31]	✔	✔	Stipple	8280	x	Bitfield	✕
FillFBWriteBufferAddr0	✕	✔	2D Set Up	8300	x	integer	✕
FillFBSourceReadBufferAddr	✕	✔	2D Set Up	8308	x	integer	✕
FillFBDestReadBufferAddr0	✕	✔	2D Set Up	8310	x	integer	✕
FillScissorMinXY	✕	✔	2D Set Up	8318	x	fixed	✕
FillScissorMaxXY	✕	✔	2D Set Up	8320	x	fixed	✕
FillForegroundColor0	✕	✔	2D Set Up	8328	x	integer	✕
FillBackgroundColor	✕	✔	2D Set Up	8330	x	integer	✕
FillConfig2D0	✕	✔	2D Set Up	8338	x	bitfield	✕
FillFBSourceReadBufferOffset	✕	✔	2D Set Up	8340	x	integer	✕
FillRectanglePosition	✕	✔	2D Set Up	8348	x	integer	✕
FillRender2D	✕	✔	2D Set Up	8350	x	bitfield	✕
FillForegroundColor1	✕	✔	2D Set Up	8358	x	integer	✕
FillConfig2D1	✕	✔	2D Set Up	8360	x	bitfield	
FillGlyphPosition	✕	✔	2D Set Up	8368	x	integer	✕
TextureCoordMode	✔	✔	Texture coord	8380	x	bitfield	✕
SStart	✔	✔	Texture Coord	8388	x	fixed	✕
dSdx	✔	✔	Texture coord	8390	x	fixed	✕
dSdyDom	✔	✔	Texture coord	8398	x	fixed	✕
TStart	✔	✔	Texture coord	83A0	x	fixed	✕
dTdx	✔	✔	Texture coord	83A8	x	fixed	✕
dTdyDom	✔	✔	Texture coord	83B0	x	fixed	✕
QStart	✔	✔	Texture Coord	83B8	x	fixed	✕
dQdx	✔	✔	Texture coord	83C0	x	fixed	✕

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
dQdyDom	✔	✔	Texture coord	83C8	x		✕
LOD	✔	✔	Texture Index	83D0	x	fixed	✕
dSdy	✔	✔	Texture coord	83D8	x	fixed	✕
dTdy	✔	✔	Texture coord	83E0	x	fixed	✕
DQdy	✔	✔	Texture coord	83E8	x	fixed	✕
S1Start	✔	✔	Texture Coord	8400	x	fixed	✕
dS1dx	✔	✔	Texture coord	8408	x	fixed	✕
dS1dyDom	✔	✔	Texture coord	8410	x	fixed	✕
T1Start	✔	✔	Texture coord	8418	x	fixed	✕
dT1dx	✔	✔	Texture coord	8420	x	fixed	✕
dT1dyDom	✔	✔	Texture coord	8428	x	fixed	✕
Q1Start	✔	✔	Texture Coord	8430	x	fixed	✕
dQ1dx	✔	✔	Texture coord	8438	x	fixed	✕
dQ1dyDom	✔	✔	Texture coord	8440	x	fixed	✕
LOD1	✔	✔	Texture Index	8448	x	fixed	✕
TextureLodBiasS	✔	✔	Texture Index	8450	x	fixed	✕
TextureLodBiasT	✔	✔	Texture Index	8458	x	fixed	✕
TextureCacheControl	✔	✔	Texture	8490	x	bitfield	✕
BorderColor0	✔	✔	Texture filter	84A8	x	bitfield	✕
LUTIndex	✔	✔	LUT	84C0	x	integer	✕
LUTData	✔	✔	LUT	84C8	x	integer	✕
LUTAddress	✔	✔	Texture Read	84D0	x	integer	✕
LUTTransfer	✔	✔	Texture Read	84D8	x	bitfield	✕
TextureFilterMode	✔	✔	Texture	84E0	x	bitfield	✕
TextureChromaUpper0	✔	✔	Color DDA	84E8	x	bitfield	✕
TextureChromaLower0	✔	✔	Color DDA	84F0	x	bitfield	✕

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
BorderColor1	✔	✔	Texture filter	84F8	x	bitfield	✕
TextureBaseAddr[16]	✔	✔	Texture Read	8500	x	integer	✕
TextureMapWidth[16]	✔	✔	Texture Read	8580	x	bitfield	✕
TextureChromaUpper1	✔	✔	Texture Filter	8600	x	bitfield	✕
TextureChromaLower1	✔	✔	Texture Filter	8608	x	bitfield	✕
TextureApplicationMode	✔	✔	Texture Application	8680	x	bitfield	✕
TextureEnvColor	✔	✔	Texture	8688	x	bitfield	✕
FogMode	✔	✔	Fog	8690	x	bitfield	✕
FogColor	✔	✔	Fog	8698	x	fixed	✕
FStart	✔	✔	Fog	86A0	x	fixed	✕
dFdx	✔	✔	Fog	86A8	x	fixed	✕
dFdyDom	✔	✔	Fog	86B0	x	fixed	✕
ZFogBias	✔	✔	Fog	86B8	x	float	✕
KsStart	✔	✔	Texture	86C8	x	fixed	✕
dKsdx	✔	✔	Texture	86D0	x	fixed	✕
dKsdyDom	✔	✔	Texture	86D8	x	fixed	✕
KdStart	✔	✔	Texture	86E0	x	fixed	✕
TextTGlyphAddr0	✕	✔	Raster-izer	8700	x	integer	✕
TextRender2DGlyph0	✕	✔	Raster-izer	8708	x	bitfield	✔
TextTGlyphAddr1	✕	✔	Raster-izer	8710	x	integer	✕
TextRender2DGlyph1	✕	✔	Raster-izer	8718	x	bitfield	✔
TextTGlyphAddr2	✕	✔	Raster-izer	8720	x	integer	✕
TextRender2DGlyph2	✕	✔	Raster-izer	8728	x	bitfield	✔
TextTGlyphAddr3	✕	✔	Raster-izer	8730	x	integer	✕
TextRender2DGlyph3	✕	✔	Raster-izer	8738	x	bitfield	✔
TextTGlyphAddr4	✕	✔	Raster-izer	8740	x	integer	✕
TextRender2DGlyph4	✕	✔	Raster-izer	8748	x	bitfield	✔
TextTGlyphAddr5	✕	✔	Raster-izer	8750	x	integer	✕
TextRender2DGlyph5	✕	✔	Raster-izer	8758	x	bitfield	✔

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
TextTGlyphAddr6	✗	✔	Raster-izer	8760	x	integer	✗
TextRender2DGlyph6	✗	✔	Raster-izer	8768	x	bitfield	✔
TextTGlyphAddr7	✗	✔	Raster-izer	8770	x	integer	✗
TextRender2DGlyph7	✗	✔	Raster-izer	8778	x	bitfield	✔
RStart	✔	✔	Color DDA	8780	x	fixed	✗
dRdx	✔	✔	Color DDA	8788	x	fixed	✗
dRdyDom	✔	✔	Color DDA Delta	8790	x	fixed	✗
GStart	✔	✔	Color DDA	8798	x	fixed	✗
dGdx	✔	✔	Color DDA	87A0	x	fixed	✗
dGdyDom	✔	✔	Color DDA	87A8	x	fixed	✗
BStart	✔	✔	Color DDA	87B0	x	fixed	✗
dBdx	✔	✔	Color DDA	87B8	x	fixed	✗
dBdyDom	✔	✔	Color DDA	87C0	x	fixed	✗
AStart	✔	✔	Color DDA	87C8	x	fixed	✗
dAdx	✔	✔	Color DDA	87D0	x	fixed	✗
dAdyDom	✔	✔	Color DDA	87D8	x	fixed	
ColorDDAMode	✔	✔	Color DDA	87E0	x	bitfield	
ConstantColor	✔	✔	Color DDA	87E8	x	bitfield	
Color	✔	✔		87F0	x	bitfield	
AlphaTestMode	✔	✔	Alpha Blend & Alpha Test	8800	x	bitfield	✗
AntialiasMode	✔	✔	Alpha test	8808	x	bitfield	✗
DitherMode	✔	✔	Dither	8818	x	bitfield	✗
FBSoftwareWriteMask	✔	✔	Logic Ops	8820	x	integer	✗
LogicalOpMode	✔	✔	Logic Op	8828	x	bitfield	✗
RouterMode	✔	✔	Router	8840	x	bitfield	✗

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
LBReadFormat	✔	✔	LB Read	8888	x	bitfield	✕
LBStencil	✕	✔	Host Out	88A8	x	bitfield	✕
LBDepth	✕	✔	LB Read/Host Out	88B0	x	integer	✕
LBWriteMode	✔	✔	LB Write	88C0	x	bitfield	✕
LBWriteFormat	✔	✔	LB Write	88C8	x	bitfield	✕
TextureData	✕	✔	Localbuffer R/W	88E8	x	integer	✕
TextureDownloadOffset	✔	✔	Localbuffer R/W	88F0	x		✕
HostInID	✔	✔	Host In	8900	x		✕
Security	✕	✔	Host In	8908	x	bitfield	✕
FlushWriteCombining	✕	✔	Host In	8910	x	integer	✕
HostInState	✔	✔	Host In	8918	x	integer	✕
HostInDMAAddress	✔	✕	Host In	8938	x	integer	✕
HostInState2	✔	✔	Host In	8940	x	integer	✕
Window	✔	✔	Stencil	8980	x	bitfield	✕
StencilMode	✔	✔	Stencil	8988	x	bitfield	✕
StencilData	✔	✔	Stencil	8990	x	bitfield	
Stencil	✔	✔	Stencil	8998	x	bitfield	✔✕
DepthMode	✔	✔	Depth	89A0	x	bitfield	✕
Depth	✔	✔	Depth	89A8	x	integer	✔✕
ZStartU	✔	✔	Depth	89B0	x	fixed	✕
ZStartL	✔	✔	Depth & Fog	89B8	x	fixed	✕
dZdxU	✔	✔	Depth & Fog	89C0	x	fixed	✕
dZdxL	✔	✔	Depth & Fog	89C8	x	fixed	✕
dZdyDomU	✔	✔	Depth & Fog	89D0	x	fixed	✕
dZdyDomL	✔	✔	Depth & Fog	89D8	x	bitfield	✕
FastClearDepth	✔	✔	Depth	89E0	x	integer	✕
FBColor	0	✕	FB Write	8A98	x	n/a	✕
FBWriteMode	✔	✔	FB Write	8AB8	x	bitfield	
FBHardwareWriteMask	✔	✔	FB Write	8AC0	x	mask	✕
FBBlockColor	✔	✕	FB Read	8AC8	x	integer	✕
FilterMode	✔	✔	Host Out	8C00	x	bitfield	✕
StatisticMode	✔	✔	Host Out	8C08	x	bitfield	✕
MinRegion	✔	✔	Host Out	8C10	x	bitfield	✕
MaxRegion	✔	✔	Host Out	8C18	x	bitfield	
ResetPickResult	✕	✔	Host Out	8C20	x	tag	✔
MinHitRegion	✕	✔	Host Out	8C28	x	bitfield	✔
MaxHitRegion	✕	✔	Host Out	8C30	x	bitfield	✔

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
PickResult	✗	✔	Host Out	8C38	x	bitfield	✔
Sync	✗	✔	Host Out	8C40	x	bitfield	✔
RLEMask	✔	✔	Host Out	8C48	x	bitfield	✗
SuspendUntilFrameBlank	✗	✔	Framebuffer Write	8C78	x	bitfield	✔
KsRStart	✔	✔	Texture Application	8C80	x	fixed	✗
dKsRdx	✔	✔	Texture	8C88	x	fixed	✗
dKsRdyDom	✔	✔	Texture	8C90	x	fixed	✗
KsGStart	✔	✔	Texture Application	8C98	x	fixed	✗
dKsGdx	✔	✔	Texture	8CA0	x	fixed	✗
dKsGdyDom	✔	✔	Texture	8CA8	x	fixed	✗
KsBStart	✔	✔	Texture Application	8CB0	x	fixed	✗
dKsBdx	✔	✔	Texture	8CB8	x	fixed	✗
dKsBdyDom	✔	✔	Texture	8CC0	x	fixed	✗
KdRStart	✔	✔	Texture	8D00	x	fixed	✗
dKdRdx	✔	✔	Texture	8D08	x	fixed	✗
dKdRdyDom	✔	✔	Texture	8D10	x	fixed	✗
KdGStart	✔	✔	Texture	8D18	x	fixed	✗
dKdGdx	✔	✔	Texture	8D20	x	fixed	✗
dKdGdyDom	✔	✔	Texture	8D28	x	fixed	✗
KdBStart	✔	✔	Texture	8D30	x	fixed	✗
dKdBdx	✔	✔	Texture	8D38	x	fixed	✗
dKdBdyDom	✔	✔	Texture	8D40	x	fixed	✗
ContextDump	✗	✔	Global	8DC0	x	bitfield	✔
ContextRestore	✗	✔	Global	8DC8	x	bitfield	✔
ContextData	✗	✔	Global	8DD0	x	bitfield	✗
LUT[0…15]	✔	✔	LUT	8E80	x	bitfield	✗
YUVMode	✔	✔	YUV Unit	8F00	x	bitfield	✗
ChromaUpper	✔	✔	Color DDA & Alpha Blend	8F08	x	bitfield	✗
ChromaLower	✔	✔	Color DDA & Alpha Blend	8F10	x	bitfield	✗
ChromaTestMode	✔	✔	Color DDA & Alpha Blend	8F18	x	bitfield	✗
FeedbackX		✔	Host Out	8F88	x	integer	✗

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
FeedbackY		✔	Host Out	8F90	x	integer	✕
EndOfFeedback	✔	✔	Host Out	8FF8	x	unused	✕
V0FloatS1	✔	✔	Delta	9000	x	float	✕
V0FloatT1	✔	✔	Delta	9008	x	float	✕
V0FloatQ1	✔	✔	Delta	9010	x	float	✕
V0FloatPackedDiffuse	✕	✔	Delta	9048	x	bitfield	✕
V0FloatKsR	✔	✔	Delta	9050	x	float	✕
V0FloatKsG	✔	✔	Delta	9058	x	float	✕
V0FloatKsB	✔	✔	Delta	9060	x	float	
V0FloatKdR	✔	✔	Delta	9068	x	float	✕
V0FloatKdG	✔	✔	Delta	9070	x	float	✕
V0FloatKdB	✔	✔	Delta	9078	x	float	✕
V1FloatS1	✔	✔	Delta	9080	x	float	✕
V1FloatT1	✔	✔	Delta	9088	x	float	✕
V1FloatQ1	✔	✔	Delta	9090	x	float	✕
V1FloatPackedDiffuse	✕	✔	Delta	90C8	x	bitfield	✕
V1FloatKsR	✔	✔	Delta	90D0	x	float	✕
V1FloatKsG	✔	✔	Delta	90D8	x	float	✕
V1FloatKsB	✔	✔	Delta	90E0	x	float	✕
V1FloatKdR	✔	✔	Delta	90E8	x	float	✕
V1FloatKdG	✔	✔	Delta	90F0	x	float	✕
V1FloatKdB	✔	✔	Delta	90F8	x	float	✕
V2FloatS1	✔	✔	Delta	9100	x	float	✕
V2FloatT1	✔	✔	Delta	9108	x	float	✕
V2FloatQ1	✔	✔	Delta	9110	x	float	✕
V2FloatPackedDiffuse	✕	✔	Delta	9148	x	bitfield	✕
V2FloatKsR	✔	✔	Delta	9150	x	float	✕
V2FloatKsG	✔	✔	Delta	9158	x	float	✕
V2FloatKsB	✔	✔	Delta	9160	x	float	✕
V2FloatKdR	✔	✔	Delta	9168	x	float	✕
V2FloatKdG	✔	✔	Delta	9170	x	float	✕
V2FloatKdB	✔	✔	Delta	9178	x	float	✕
V0FloatS	✔	✔	Delta	9180	x	float	✕
V0FloatT	✔	✔	Delta	9188	x	float	✕
V0FloatQ	✔	✔	Delta	9190	x	float	✕
V0FloatR	✔	✔	Delta	91A8	x	float	✕
V0FloatG	✔	✔	Delta	91B0	x	float	✕
V0FloatB	✔	✔	Delta	91B8	x	float	✕
V0FloatA	✔	✔	Delta	91C0	x	float	✕
V0FloatF	✔	✔	Delta	91C8	x	float	✕
V0FloatX	✔	✔	Delta	91D0	x	float	✕
V0FloatY	✔	✔	Delta	91D8	x	float	✕
V0FloatZ	✔	✔	Delta	91E0	x	float	✕
V0FloatPackedColor	✕	✔	Delta	91F0	x	bitfield	✕

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
V0FloatPackedSpecularFog	✗	✔	Delta	91F8	x	bitfield	✗
V1FloatS	✔	✔	Delta	9200	x	float	✗
V1FloatT	✔	✔	Delta	9208	x	float	✗
V1FloatQ	✔	✔	Delta	9210	x	float	✗
V1FloatR	✔	✔	Delta	9228	x	float	✗
V1FloatG	✔	✔	Delta	9230	x	float	✗
V1FloatB	✔	✔	Delta	9238	x	float	✗
V1FloatA	✔	✔	Delta	9240	x	float	✗
V1FloatF	✔	✔	Delta	9248	x	float	✗
V1FloatX	✔	✔	Delta	9250	x	float	✗
V1FloatY	✔	✔	Delta	9258	x	float	✗
V1FloatZ	✔	✔	Delta	9260	x	float	✗
V1FloatPackedColor	✗	✔	Delta	9270	x	bitfield	✗
V1FloatPackedSpecularFog	✗	✔	Delta	9278	x	bitfield	✗
V2FloatS	✔	✔	Delta	9280	x	float	✗
V2FloatT	✔	✔	Delta	9288	x	float	✗
V2FloatQ	✔	✔	Delta	9290	x	float	✗
V2FloatR	✔	✔	Delta	92A8	x	float	✗
V2FloatG	✔	✔	Delta	92B0	x	float	✗
V2FloatB	✔	✔	Delta	92B8	x	float	✗
V2FloatA	✔	✔	Delta	92C0	x	float	✗
V2FloatF	✔	✔	Delta	92C8	x	float	✗
V2FloatX	✔	✔	Delta	92D0	x	float	✗
V2FloatY	✔	✔	Delta	92D8	x	float	✗
V2FloatZ	✔	✔	Delta	92E0	x	float	✗
V2FloatPackedColor	✗	✔	Delta	92F0	x	bitfield	✗
V2FloatPackedSpecularFog	✗	✔	Delta	92F8	x	bitfield	✗
DeltaMode	✔	✔	Delta	9300	x	bitfield	✗
DrawTriangle	✗	✔	Delta	9308	x	bitfield	✔
RepeatTriangle	✗	✔	Delta	9310	x	tag	✔
DrawLine0	✗	✔	Delta	9318	x	fixed	✔
DrawLine1	✗	✔	Delta	9320	x	fixed	✔
RepeatLine	✗	✔	Delta	9328	x	tag	✔
DrawPoint	✗	✔	Delta	9330	x	bitfield	✔
ProvokingVertex	✔	✔	Delta	9338	x	bitfield	✔
TextureLODScale	✔	✔	Texture coord	9340	x	float	✗
TextureLODScale1	✔	✔	Texture coord	9348	x	float	✗
DeltaControl	✔	✔	Delta	9350	x	bitfield	✗
ProvokingVertexMask	✔	✔	Delta	9358	x	bitfield	✗
XBias	✔	✔	Delta	9480	x	float	✗
YBias	✔	✔	Delta	9488	x	float	✗
PointSize	✔	✔	Delta	9498	x	integer	✗

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
AAPointSize	✔	✔	Delta	94A0	x	float	✕
LineMode	✔	✔	Delta	94A8	x	bitfield	✕
LineWidth	✔	✔	Delta	94B0	x	integer	✕
LineWidthOffset	✔	✔	Delta	94B8	x	integer	✕
AALineWidth	✔	✔	Delta	94C0	x	float	✕
RectangleHeight	✔	✔	Delta	94E0	x	float	✕
LineCoord0	✕	✔	Delta	9760	x	bitfield	✕
DrawLine2D10	✕	✔	Delta	9768	x	bitfield	✔
LineCoord1	✕	✔	Delta	9770	x	bitfield	✕
DrawLine2D01	✕	✔	Delta	9778	x	bitfield	✔
DMAAddr	✕	✔	Host In	A980	x	integer	✕
DMACount	✕	✔	Host In	A988	x	integer	✕
Command Interrupt	✕	✔	Host In	A990	x	bitfield	✕
DMARectangle Read	✕	✔	Host In	A9A8	x	bitfield	✕
DMARectangleReadAddress	✔	✔	Host In	A9B0	x	integer	✕
DMARectangleRead LinePitch	✔	✔	Host In	A9B8	x	integer	✕
DMARectangleRead Target	✔	✔	Host In	A9C0	x	bitfield	✕
DMARectangleWrite	✕	✔	Host In	A9C8	x	bitfield	✕
DMARectangleWrite Address	✔	✔	Host In	A9D0	x	integer	✕
DMARectangleWriteLinePitch	✔	✔	Host In	A9D8	x	integer	✕
DMAOutput Address	✕	✔	Host In	A9E0	x	integer	✕
DMAOutputCount	✕	✔	Host In	A9E8	x	integer	✕
DMAContinue	✕	✔	Host In	A9F8	x	integer	✔
DMAFeedback	✕	✔	Host In	AA10	x	integer	✕
DeltaModeAnd	✕	✔	Delta	AAD0	x	bitfield	✕
DeltaModeOr	✕	✔	Delta	AAD8	x	bitfield	✕
LineModeAnd	✕	✔	Delta	AAF0	x	bitfield	✕
LineModeOr	✕	✔	Delta	AAF8	x	bitfield	✕
DeltaControlAnd	✕	✔	Delta	AB20	x	bitfield	✕
DeltaControlOr	✕	✔	Delta	AB28	x	bitfield	✕
WindowAnd	✕	✔	Stencil	AB80	x	bitfield	✕
WindowOr	✕	✔	Stencil	AB88	x	bitfield	✕
RasterizerModeAnd	✕	✔	Raster-izer	ABA0	x	bitfield	✕
RasterizerModeOr	✕	✔	Raster-izer	ABA8	x	bitfield	✕
ScissorModeAnd	✕	✔	Scissor	ABB0	x	bitfield	✕
ScissorModeOr	✕	✔	Scissor	ABB8	x	bitfield	✕
LineStippleModeAnd	✕	✔	Stipple	ABC0	x	bitfield	✕
LineStippleModeOr	✕	✔	Stipple	ABC8	x	bitfield	✕
AreaStippleModeAnd	✕	✔	Stipple	ABD0	x	bitfield	✕
AreaStippleModeOr	✕	✔	Stipple	ABD8	x	bitfield	✕
ColorDDAModeAnd	✕	✔	Color DDA	ABE0	x	bitfield	✕

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
ColorDDAModeOr	✗	✔	Color DDA	ABE8	x	bitfield	✗
AlphaTestModeAnd	✗	✔	Alpha Blend & Alpha Test	ABF0	x	bitfield	✗
AlphaTestModeOr	✗	✔	Alpha Blend & Alpha Test	ABF8	x	bitfield	✗
AntialiasModeAnd	✗	✔	Alpha test	AC00	x	bitfield	✗
AntialiasModeOr	✗	✔	Alpha test	AC08	x	bitfield	✗
FogModeAnd	✗	✔	Fog	AC10	x	bitfield	✗
FogModeOr	✗	✔	Fog	AC18	x	bitfield	✗
TextureCoordModeAnd	✗	✔	Texture coord	AC20	x		✗
TextureCoordModeOr	✗	✔	Texture coord	AC28	x	bitfield	✗
TextureReadMode0And	✗	✔	Texture Read	AC30	x	bitfield	✗
TextureReadMode0Or	✗	✔	Texture Read	AC38	x	bitfield	✗
TextureApplication ModeAnd	✗	✔	Texture Applicatio n	AC50	x	bitfield	✗
TextureApplication ModeOr	✗	✔	Texture Applicatio n	AC58	x	bitfield	✗
StencilModeAnd	✗	✔	Stencil	AC60	x	bitfield	✗
StencilModeOr	✗	✔	Stencil	AC68	x	bitfield	✗
DepthModeAnd	✗	✔	Depth	AC70	x	bitfield	✗
DepthModeOr	✗	✔	Depth	AC78	x	bitfield	✗
LBWriteModeAnd	✗	✔	LB Write	AC80	x	bitfield	✗
LBWriteModeOr	✗	✔	LB Write	AC88	x	bitfield	✗
FBDestReadModeAnd	✗	✔	FB Read	AC90	x	bitfield	✗
FBDestReadModeOr	✗	✔	FB Read	AC98	x	bitfield	✗
FBSourceReadModeAnd	✗	✔	FB Read	ACA0	x	bitfield	✗
FBSourceReadModeOr	✗	✔	FB Read	ACA8	x	bitfield	✗
AlphaBlendColorModeAnd	✗	✔	Alpha blend	ACB0	x	bitfield	✗
AlphaBlendColorModeOr	✗	✔	Alpha blend	ACB8	x	bitfield	✗
ChromaTestModeAnd	✗	✔	Color DDA & Alpha Blend	ACC0	x	bitfield	✗

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
ChromaTestModeOr	✗	✔	Color DDA & Alpha Blend	ACC8	x	bitfield	✗
DitherModeAnd	✗	✔	Dither	ACD0	x	bitfield	✗
DitherModeOr	✗	✔	Dither	ACD8	x	bitfield	✗
LogicalOpModeAnd	✗	✔	Logic Op	ACE0	x	bitfield	✗
LogicalOpModeOr	✗	✔	Logic Op	ACE8	x	bitfield	✗
FBWriteModeAnd	✗	✔	FB Write	ACF0	x	bitfield	✗
FBWriteModeOr	✗	✔	FB Write	ACF8	x	bitfield	✗
FilterModeAnd	✗	✔	Host Out	AD00	x	bitfield	✗
FilterModeOr	✗	✔	Host Out	AD08	x	bitfield	✗
StatisticModeAnd	✗	✔	Host Out	AD10	x	bitfield	✗
StatisticModeOr	✗	✔	Host Out	AD18	x	bitfield	✗
FBDestReadEnablesAnd	✗	✔	FB Read	AD20	x	bitfield	✗
FBDestReadEnablesOr	✗	✔	FB Read	AD28	x	bitfield	✗
AlphaBlendAlphaModeAnd	✗	✔	Alpha blend	AD30	x	bitfield	✗
AlphaBlendAlphaModeOr	✗	✔	Alpha blend	AD38	x	bitfield	✗
TextureReadMode1And	✗	✔	Texture Read	AD40	x	bitfield	✗
TextureReadMode1Or	✗	✔	Texture Read	AD48	x	bitfield	✗
TextureFilterModeAnd	✗	✔	Texture	AD50	x	bitfield	✗
TextureFilterModeOr	✗	✔	Texture	AD58	x	bitfield	✗
LUTModeAnd	✗	✔	LUT	AD70	x	bitfield	✗
LUTModeOr	✗	✔	LUT	AD78	x	bitfield	✗
Zstart	✔	✔	Fog	ADD8	x	integer	✗
FBDestReadBufferAddr[0…3]	✔	✔	FB Read	AE80	x	integer	✗
FBDestReadBufferOffset[0…3]	✔	✔	FB Read	AEA0	x	integer	✗
FBDestReadBufferWidth[0…3]	✔	✔	FB Read	AEC0	x	integer	✗
FBDestReadMode	✔	✔	FB Read	AEE0	x	bitfield	✗
FBDestReadEnables	✔	✔	FB Read	AEE8	x	bitfield	✗
FBSourceReadMode	✔	✔	FB Read	AF00	x	bitfield	✗
FBSourceReadBufferAddr	✔	✔	FB Read	AF08	x	integer	✗
FBSourceReadBufferOffset	✔	✔	FB Read	AF10	x	integer	✗
FBSourceReadBufferWidth	✔	✔	FB Read	AF18	x	integer	✗
AlphaSourceColor	✔	✔	Alpha blend	AF80	x	integer	✗
AlphaDestColor	✔	✔	Alpha blend	AF88	x	bitfield	✗
ChromaPassColor	✔	✔	Color DDA & Alpha Blend	AF90	x	bitfield	✗

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
ChromaFailColor	✔	✔	Color DDA & Alpha Blend	AF98	x	bitfield	✕
AlphaBlendColorMode	✔	✔	Alpha blend	AFA0	x	bitfield	✕
AlphaBlendAlphaMode	✔	✔	Alpha blend	AFA8	x	bitfield	✕
ConstantColorDDA	✕	✔	Color DDA	AFB0	x	bitfield	✕
FBWriteBufferAddr[0…3]	✔	✔	FB Write	B000	x	integer	✕
FBWriteBufferOffset[0…3]	✔	✔	FB Write	B020	x	integer	✕
FBWriteBufferWidth[0…3]	✔	✔	FB Write	B040	x	integer	✕
FBBlockColor[0…3]	✔	✔	FB Write	B060	x	integer	✕
FBBlockColorBack[0…3]	✔	✔	FB Write	B080	x	integer	✕
FBBlockColorBack	✔	✔	FB Write	B0A0	x	integer	✕
SizeOfFramebuffer	✔	✔	LB Read, FB Read, FB Write	B0A8	x	integer	✕
VTGAddress	✔	✔	FB Write	B0B0	x	integer	✔
VTGData	✔	✔	FB Write	B0B8	x	integer	✔
ForegroundColor	✔	✔	Logic Ops	B0C0	x	integer	✕
BackgroundColor	✔	✔	Logic Ops	B0C8	x	integer	✕
FogTable[0…15]	✔	✔	Fog	B100	x	bitfield	✕
FogTable[16…31]	✔	✔	Fog	B180	x	bitfield	✕
FogTable[32…47]	✔	✔	Fog	B200	x	bitfield	✕
FogTable[48…63]	✔	✔	Fog	B280	x	bitfield	✕
TextureCompositeMode	✔	✔	Texture Composite	B300	x	bitfield	✕
TextureCompositeColorMode0	✔	✔	Texture Composite	B308	x	bitfield	✕
TextureCompositeAlphaMode0	✔	✔	Texture Composite	B310	x	bitfield	✕
TextureCompositeColorMode1	✔	✔	Texture Composite	B318	x	bitfield	✕
TextureCompositeAlphaMode1	✔	✔	Texture Composite	B320	x		✕
TextureCompositeFactor0	✔	✔	Texture Composite	B328	x	bitfield	

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
TextureCompositeFactor1	✔	✔	Texture Composite	B330	x	bitfield	✕
TextureIndexMode0	✔	✔	Texture Index	B338	x	bitfield	✕
TextureIndexMode1	✔	✔	Texture Index	B340	x	bitfield	✕
LodRange0	✔	✔	Texture Index	B348	x	bitfield	✕
LodRange1	✔	✔	Texture Index	B350	x	fixed	✕
InvalidateCache	✕	✔	Texture Read	B358	x	bitfield	✔
SetLogicalTexturePage	✔	✔	Texture Read	B360	x	bitfield	✕
UpdateLogicalTextureInfo	✕	✔	Texture Read	B368	x	tag	✔
TouchLogicalPage	✕	✔	Texture Read	B370	x	bitfield	✔
LUTMode	✔	✔	LUT	B378	x	bitfield	✕
TextureCompositeColorMode0And	✕	✔	Texture Composite	B380	x	bitfield	✕
TextureCompositeColorMode0Or	✕	✔	Texture Composite	B388	x	bitfield	✕
TextureCompositeAlphaMode0And	✕	✔	Texture Composite	B390	x	bitfield	✕
TextureCompositeAlphaMode0Or	✕	✔	Texture Composite	B398	x	bitfield	✕
TextureCompositeColorMode1And	✕	✔	Texture Composite	B3A0	x	bitfield	✕
TextureCompositeColorMode1Or	✕	✔	Texture Composite	B3A8	x	bitfield	✕
TextureCompositeAlphaMode1And	✕	✔	Texture Composite	B3B0	x	bitfield	✕
TextureCompositeAlphaMode1Or	✕	✔	Texture Composite	B3B8	x	bitfield	✕
TextureIndexMode0And	✕	✔	Texture Index	B3C0	x	bitfield	✕
TextureIndexMode0Or	✕	✔	Texture Index	B3C8	x	bitfield	✕

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
TextureIndexMode1And	✗	✔	Texture Index	B3D0	x	bitfield	✗
TextureIndexMode1Or	✗	✔	Texture Index	B3D8	x	bitfield	✗
StencilDataAnd	✗	✔	Stencil	B3E0	x	bitfield	✗
StencilDataOr	✗	✔	Stencil	B3E8	x	bitfield	✗
TextureReadMode0	✔	✔	Texture Read	B400	x	bitfield	✗
TextureReadMode1	✔	✔	Texture Read	B408	x	bitfield	✗
TextureMapSize	✔	✔	Texture Read	B428	x	integer	✗
HeadPhysicalPage Allocation[0…3]	✔	✔	Texture Read	B480	x	integer	✗
TailPhysicalPage Allocation[0…3]	✔	✔	Texture Read	B4A0	x	integer	✗
PhysicalPageAllocationTableAddr	✔	✔	Texture Read	B4C0	x	integer	✗
BasePageOfWorking Set	✔	✔	Texture Read	B4C8	x	integer	✗
LogicalTexturePage TableAddr	✔	✔	Texture Read	B4D0	x	integer	✗
LogicalTexturePage TableLength	✔	✔	Texture Read	B4D8	x	integer	✗
BasePageOfWorking SetHost	✔	✔	Texture Read	B4E0	x	integer	✗
LBDestReadMode	✔	✔	LB Read	B500	x	integer	✗
LBDestReadEnables	✔	✔	LB Read	B508	x	bitfield	✗
LBDestReadBufferAddr	✔	✔	LB Read	B510	x	integer	
LBDestReadBufferOffset	✔	✔	LB Read	B518	x	integer	
LBSourceReadMode	✔	✔	LB Read	B520	x	integer	✗
LBSourceReadBufferAddr	✔	✔	LB Read	B528	x	integer	✗
LBSourceReadBufferOffset	✔	✔	LB Read	B530	x	bitfield	✗
GIDMode	✔	✔	LB Read	B538	x	bitfield	✗
LBWriteBufferAddr	✔	✔	LB Write	B540	x	integer	✗
LBWriteBufferOffset	✔	✔	LB Write	B548	x	integer	✗
LBClearDataL	✔	✔	LB Read	B550	x	integer	✗
LBClearDataU	✔	✔	LB Read	B558	x	integer	✗
LBDestReadModeAnd	✗	✔	LB Read	B580	x	bitfield	✗
LBDestReadModeOr	✗	✔	LB Read	B588	x	bitfield	✗
LBDestReadEnables And	✗	✔	LB Read	B590	x	bitfield	✗
LBDestReadEnables Or	✗	✔	LB Read	B598	x	bitfield	✗
LBSourceReadMode And	✗	✔	LB Read	B5A0	x	bitfield	✗
LBSourceReadModeOr	✗	✔	LB Read	B5A8	x	bitfield	✗
GIDModeAnd	✗	✔	LB Read	B5B0	x	bitfield	✗
GIDModeOr	✗	✔	LB Read	B5B8	x	bitfield	✗

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
RectanglePosition	✔	✔	2D Set Up	B600	x	integer	✕
GlyphPosition	✔	✔	2D Set Up	B608	x	integer	✕
RenderPatchOffset	✔	✔	2D Set Up	B610	x	bitfield	✕
Config2D	✕	✔	Global	B618	x	bitfield	✕
Packed8Pixels	✕	✔	2D Set Up	B630	x	integer	✔
Packed16Pixels	✕	✔	2D Set Up	B638	x	integer	✔
Render2D	✕	✔	2D Set Up	B640	x	bitfield	✕
Render2DGlyph	✕	✔	2D Set Up	B648	x	bitfield	✕
DownloadTarget	✔	✔	2D Set Up	B650	x		✔
DownloadGlyphWidth	✔	✔	2D Set Up	B658	x	integer	✕
GlyphData	✕	✔	2D Set Up	B660	x	integer	✕
Packed4Pixels	✕	✔	2D Set Up	B668	x	integer	✔
RLData	✔	✔	2D Set Up	B670	x	integer	✕
RLCount	✕	✔	2D Set Up	B678	x	integer	✕
IndexBaseAddress	✔	✔	Host In	B700	x	integer	✕
VertexBaseAddress	✔	✔	Host In	B708	x	integer	✕
IndexedTriangleList	✕	✔	Host In	B710	x	integer	✕
IndexedTriangleFan	✕	✔	Host In	B718	x	integer	✕
IndexedTriangleStrip	✕	✔	Host In	B720	x	integer	✕
IndexedLineList	✕	✔	Host In	B728	x	integer	✕
IndexedLineStrip	✕	✔	Host In	B730	x	integer	✕
IndexedPointList	✕	✔	Host In	B738	x	integer	✕
IndexedPolygon	✕	✔	Host In	B740	x	integer	✕
VertexTriangleList	✕	✔	Host In	B748	x	integer	✕
VertexTriangleFan	✕	✔	Host In	B750	x	integer	✕
VertexTriangleStrip	✕	✔	Host In	B758	x	integer	✕
VertexLineList	✕	✔	Host In	B760	x	integer	✕
VertexLineStrip	✕	✔	Host In	B768	x	integer	✕
VertexPointList	✕	✔	Host In	B770	x	integer	✕
VertexPolygon	✕	✔	Host In	B778	x	integer	✕
DMAMemoryControl	✔	✔	Host In	B780	x	bitfield	✕
VertexValid	✔	✔	Host In	B788	x	integer	✕
VertexFormat	✔	✔	Host In	B790	x	integer	✕
VertexControl	✔	✔	Host In	B798	x	bitfield	✕

Name	Read-back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
RetainedRender	✔	✔	Host In	B7A0	x	bitfield	✔
IndexedVertex	✕	✔	Host In	B7A8	x	integer	✕
IndexedDoubleVertex	✕	✔	Host In	B7B0	x	integer	✕
Vertex0	✕	✔	Host In	B7B8	x	integer	✕
Vertex1	✕	✔	Host In	B7C0	x	integer	✕
Vertex2	✕	✔	Host In	B7C8	x	integer	✕
VertexData0	✕	✔	Host In	B7D0	x	integer	✕
VertexData1	✕	✔	Host In	B7D8	x	integer	✕
VertexData2	✕	✔	Host In	B7E0	x	integer	✕
VertexData	✕	✔	Host In	B7E8	x	integer	✕
VertexTagList[0…15]	✔	✔	Host In	B800	x	bitfield	✕
VertexTagList[16…31]	✔	✔	Host In	B880	x	bitfield	✕

7

Package Diagrams

The package is a standard 456 ball PBGA. The chamfered corner indicates pin 1AF.

Figure 7-1 Package Diagram (Bottom View)

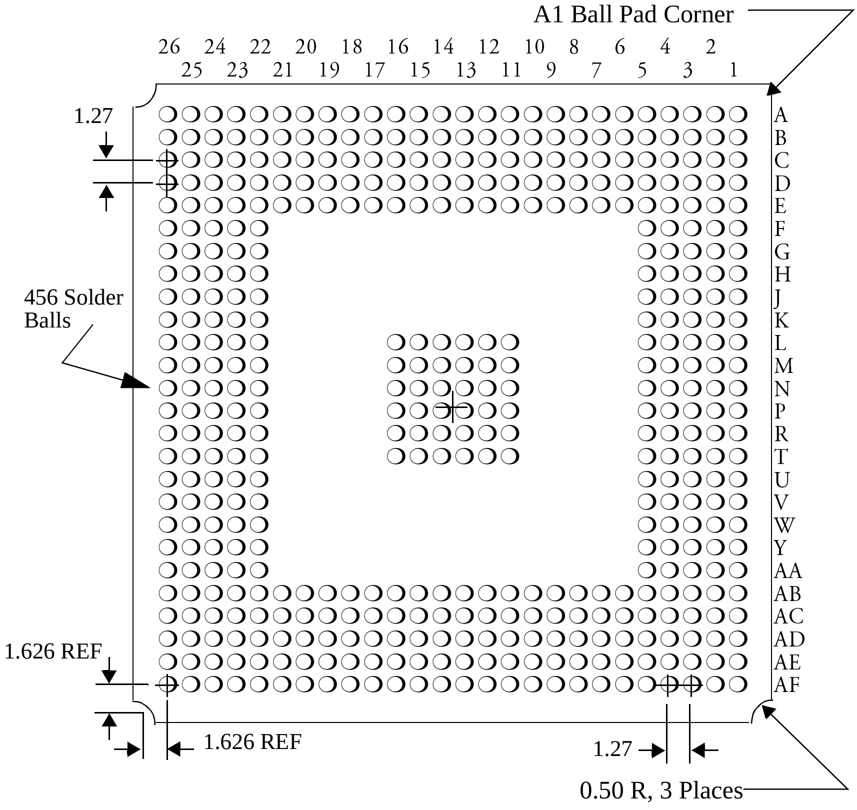


Figure 7-2 Package Diagram (Top View)

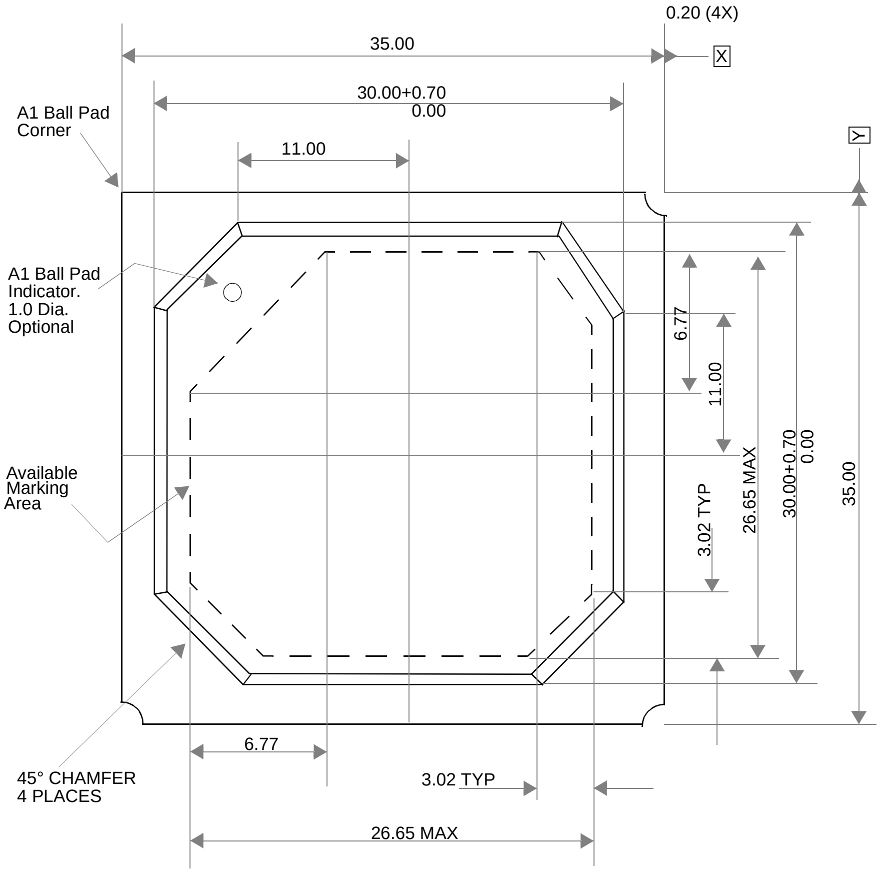


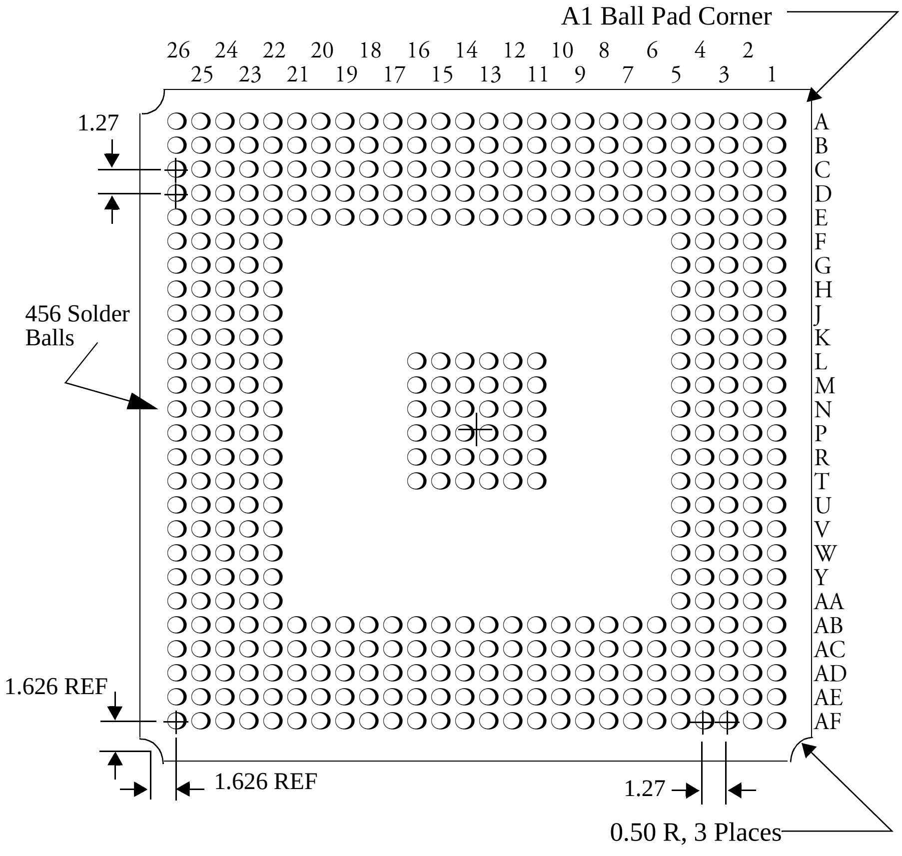
Table 7-1 Mechanical Diagrams

Dimension		mm
A)	Ball Pitch	1.27
B)	Lead Width	0.63 ±0.03
C)	Height	2.33
D)	Body Width	35

8

Pin Assignment

Figure 8-1 Package Diagram (Bottom View)



8.1 Pinlist by Number

The table below provides a brief description of each pin. It is organized alphabetically by pin number.

The pin type definitions used are:

- I/O: Input Signal
- GND: Ground
- VSS 3.3: Power at 3.3V
- VSS 2.5: Power at 2.5 Volts

Where AGP pins are unused, the following terminations are recommended:

AGPSBA(7:0)	No connection - output only
AGPPipeN	No connection - output only
AGPADSTB(1:0)	Tie high (input only)
AGPADSTBN(1:0)	Tie low (input only)
AGPADSTB	No connection - output only
AGPADSTBN	No connection - output only
AGPADSt(2:0)	Tie high - input only
AGPVREF	As per AGP termination
AGPRbfN	No connection (output only)

Table 8-1 Pinlist by Number

NO.	NAME	DESCRIPTION
A1	VCC_2.5	
A2	VCC_2.5	
A3	MDat_124	Memory data line 124
A4	MDat_98	Memory data line 98
A5	MDat_113	Memory data line 113
A6	MDat_112	Memory data line 112
A7	MDat_119	Memory data line 119
A8	MDat_111	Memory data line 111
A9	MDat_104	Memory data line 104
A10	MemClkRet_3	Memory Clock Return 3

Table 8-1 Pinlist by Number

NO.	NAME	DESCRIPTION
A11	MDat_90	Memory data line 90
A12	MDat_91	Memory data line 91
A13	MDat_70	Memory data line 70
A14	MDat_82	Memory data line 82
A15	MDat_83	Memory data line 83
A16	MDat_77	Memory data line 77
A17	MDat_76	Memory data line 76
A18	MDat_75	Memory data line 75
A19	MemClkRet_2	Memory Clock Return 2
A20	MDat_60	Memory data line 60
A21	MDat_59	Memory data line 59
A22	MDat_35	Memory data line 35
A23	MDat_36	Memory data line 36
A24	MDat_38	Memory data line 38
A25	VCC_2.5	
A26	VCC_2.5	
B1	VCC_2.5	
B2	VCC_2.5	
B3	MDat_99	Memory data line 99
B4	MDat_97	Memory data line 97
B5	MDat_115	Memory data line 115
B6	MDat_116	Memory data line 116
B7	MDat_108	Memory data line 108
B8	MDat_107	Memory data line 107
B9	MByte_15	Memory byte select 15
B10	MDat_93	Memory data line 93
B11	MDat_89	Memory data line 89
B12	MDat_66	Memory data line 66
B13	MDat_69	Memory data line 69
B14	MDat_81	Memory data line 81

Table 8-1 Pinlist by Number

NO.	NAME	DESCRIPTION
B15	MDat_85	Memory data line 85
B16	MDat_78	Memory data line 78
B17	MDat_74	Memory data line 74
B18	MByte_11	Memory byte select 11
B19	MByte_9	Memory byte select 9
B20	MDat_61	Memory data line 61
B21	MDat_58	Memory data line 58
B22	MDat_34	Memory data line 34
B23	MDat_37	Memory data line 37
B24	MDat_39	Memory data line 39
B25	VCC_2.5	
B26	VCC_2.5	
C1	MDat_123	Memory data line 123
C2	MDat_125	Memory data line 125
C3	VCC_2.5	
C4	MDat_96	Memory data line 96
C5	MDat_114	Memory data line 114
C6	MDat_117	Memory data line 117
C7	MDat_109	Memory data line 109
C8	MDat_106	Memory data line 106
C9	MByte_13	Memory byte select 13
C10	MDat_94	Memory data line 94
C11	MDat_88	Memory data line 88
C12	MDat_65	Memory data line 65
C13	MDat_67	Memory data line 67
C14	MDat_80	Memory data line 80
C15	MDat_86	Memory data line 86
C16	MDat_79	Memory data line 79
C17	MDat_73	Memory data line 73
C18	MByte_8	Memory byte select 8

Table 8-1 Pinlist by Number

NO.	NAME	DESCRIPTION
C19	MByte_10	Memory byte select 10
C20	MDat_62	Memory data line 62
C21	MDat_57	Memory data line 57
C22	MDat_33	Memory data line 33
C23	MDat_48	Memory data line 48
C24	VCC_2.5	
C25	MDat_49	Memory data line 49
C26	MDat_50	Memory data line 50
D1	MDat_122	Memory data line 122
D2	MDat_100	Memory data line 100
D3	MDat_126	Memory data line 126
D4	VCC_2.5	
D5	MDat_118	Memory data line 118
D6	MDat_110	Memory data line 110
D7	MDat_105	Memory data line 105
D8	MByte_12	Memory byte select 12
D9	MByte_14	Memory byte select 14
D10	MDat_95	Memory data line 95
D11	MDat_92	Memory data line 92
D12	MDat_64	Memory data line 64
D13	MDat_71	Memory data line 71
D14	MDat_68	Memory data line 68
D15	MDat_87	Memory data line 87
D16	MDat_84	Memory data line 84
D17	MDat_72	Memory data line 72
D18	RenderSyncN	Multi-rasterizer i/o sync
D19	MDat_63	Memory data line 63
D20	MDat_56	Memory data line 56
D21	MDat_32	Memory data line 32
D22	VideoExtCtrl	Video External control

Table 8-1 Pinlist by Number

NO.	NAME	DESCRIPTION
D23	VCC_2.5	
D24	MDat_51	Memory data line 51
D25	MDat_52	Memory data line 52
D26	MDat_53	Memory data line 53
E1	MDat_121	Memory data line 121
E2	MDat_101	Memory data line 101
E3	MDat_102	Memory data line 102
E4	MDat_127	Memory data line 127
E5	GND	
E6	GND	
E7	VCC_3.3	
E8	VCC_3.3	
E9	GND	
E10	GND	
E11	VCC_3.3	
E12	VCC_3.3	
E13	GND	
E14	GND	
E15	VCC_3.3	
E16	VCC_3.3	
E17	GND	
E18	GND	
E19	VCC_3.3	
E20	VCC_3.3	
E21	GND	
E22	GND	
E23	MDat_47	Memory data line 47
E24	MDat_46	Memory data line 46
E25	MDat_55	Memory data line 55
E26	MDat_54	Memory data line 54

Table 8-1 Pinlist by Number

NO.	NAME	DESCRIPTION
F1	MDat_120	Memory data line 120
F2	VSBResetN	Video Stream B Reset Out
F3	MDat_103	Memory data line 103
F4	VSBData_0	VideoStream B data line 0
F5	GND	
F22	GND	
F23	MDat_45	Memory data line 45
F24	MDat_44	Memory data line 44
F25	MDat_43	Memory data line 43
F26	MDat_42	Memory data line 42
G1	VSBData_1	VideoStream B data line 1
G2	VSBData_2	VideoStream B data line 2
G3	VSBData_3	VideoStream B data line 3
G4	VSBData_4	VideoStream B data line 4
G5	VCC_3.3	
G22	VCC_3.3	
G23	MByte_4	Memory byte select 4
G24	MByte_7	Memory byte select 7
G25	MDat_40	Memory data line 40
G26	MDat_41	Memory data line 41
H1	VSBData_5	VideoStream B data line 5
H2	VSBData_6	VideoStream B data line 6
H3	VSBData_7	VideoStream B data line 7
H4	VSBClk	VideoStream B clock
H5	VCC_3.3	
H22	VCC_3.3	
H23	MDat_31	Memory data line 31
H24	MByte_6	Memory byte select 6
H25	MByte_5	Memory byte select 5
H26	MemClkRet_1	Memory Clock Return 1

Table 8-1 Pinlist by Number

NO.	NAME	DESCRIPTION
J1	VSGPDataStrob eN	VS GP bus data strobe
J2	VSGPReadWrit eN	VS GP bus read/write signal
J3	VSBClkOut	Video Streams B Clock Out
J4	SPARE	
J5	GND	
J22	GND	
J23	MDat_30	Memory data line 30
J24	MDat_29	Memory data line 29
J25	MDat_28	Memory data line 28
J26	MBank_3	Memory bank select 3
K1	VSCtl_0	VideoStreams Control line 0
K2	VSCtl_1	VideoStreams Control line 1
K3	VSGPChipSelectN	VS GP bus chip select
K4	VSGPDataAckN	VS GP bus data ack
K5	GND	
K22	GND	
K23	MDat_24	Memory data line 24
K24	MDat_25	Memory data line 25
K25	MDat_26	Memory data line 26
K26	MDat_27	Memory data line 27
L1	VSCtl_3	VideoStreams Control line 3
L2	VSCtl_4	VideoStreams Control line 4
L3	VSCtl_5	VideoStreams Control line 5
L4	VSCtl_2	VideoStreams Control line 2
L5	VCC_3.3	
L11	GND	
L12	GND	
L13	GND	
L14	GND	
L15	GND	

Table 8-1 Pinlist by Number

NO.	NAME	DESCRIPTION
L16	GND	
L22	VCC_3.3	
L23	MDat_3	Memory data line 3
L24	MDat_0	Memory data line 0
L25	MDat_1	Memory data line 1
L26	MDat_2	Memory data line 2
M1	TestSel_0_	Test Mode Select 1
M2	TestSel_2_	Test Mode Select 2
M3	VSCtl_6	VideoStreams Control line 6
M4	VSCtl_7	VideoStreams Control line 7
M5	VCC_3.3	
M11	GND	
M12	GND	
M13	GND	
M14	GND	
M15	GND	
M16	GND	
M22	VCC_3.3	
M23	MDat_7	Memory data line 7
M24	MDat_6	Memory data line 6
M25	MDat_5	Memory data line 5
M26	MDat_4	Memory data line 4
N1	DacAVDD	Analog/video DAC
N2	vidRed	Analog red signal
N3	vidGreen	Analog green signal
N4	VidRightEye	Right signal for stereo
N5	GND	
N11	GND	
N12	GND	
N13	GND	

Table 8-1 Pinlist by Number

NO.	NAME	DESCRIPTION
N14	GND	
N15	GND	
N16	GND	
N22	GND	
N23	MDat_16	Memory data line 16
N24	MDat_19	Memory data line 19
N25	MDat_18	Memory data line 18
N26	MDat_17	Memory data line 17
P1	VidVRef	Voltage reference
P2	DacComp	Compensation pin
P3	vidBlue	Analog blue signal
P4	vidResRef	Reference resistor
P5	GND	
P11	GND	
P12	GND	
P13	GND	
P14	GND	
P15	GND	
P16	GND	
P22	GND	
P23	MDat_20	Memory data line 20
P24	MDat_23	Memory data line 23
P25	MDat_22	Memory data line 22
P26	MDat_21	Memory data line 21
R1	TestMode	Test Mode control
R2	VidHSync	Horizontal sync
R3	TestSel_1	Test Mode Select 1
R4	DacAGnd	DAC Power/Gnd pin
R5	VCC_3.3	
R11	GND	

Table 8-1 Pinlist by Number

NO.	NAME	DESCRIPTION
R12	GND	
R13	GND	
R14	GND	
R15	GND	
R16	GND	
R22	VCC_3.3	
R23	MDat_15	Memory data line 15
R24	MDat_14	Memory data line 14
R25	MDat_13	Memory data line 13
R26	MDat_12	Memory data line 12
T1	Xtal1	Crystal i/p 1
T2	VidVsync	Vertical sync
T3	PLLDISABLE	PLL Disable
T4	Xtal2	Crystal i/p 2
T5	VCC_3.3	
T11	GND	
T12	GND	
T13	GND	
T14	GND	
T15	GND	
T16	GND	
T22	VCC_3.3	
T23	MDat_11	Memory data line 11
T24	MDat_8	Memory data line 8
T25	MDat_9	Memory data line 9
T26	MDat_10	Memory data line 10
U1	ROMWeN	ROM Write Enable
U2	ROMSelN	ROM Select signal
U3	VidDDCData	Data line for DDC
U4	VidDDCClk	Clock line for DDC

Table 8-1 Pinlist by Number

NO.	NAME	DESCRIPTION
U5	GND	
U22	GND	
U23	MBank_2	Memory bank select 2
U24	MByte_0	Memory byte select 0
U25	MByte_3	Memory byte select 3
U26	MByte_1	Memory byte select 1
V1	SBClk	Serial bus clock
V2	SPARE	
V3	VSAClk	VideoStream A clock
V4	VSAResetN	Video Stream reset
V5	GND	
V22	GND	
V23	MBank_0	Memory bank select 0
V24	MBank_1	Memory bank select 1
V25	MByte_2	Memory byte select 2
V26	MemClkRet_0	Memory Clock Return 0
W1	PLLPower	PLL Power/Gnd pin
W2	VSAData_5	VideoStream A data line 5
W3	VSAData_7	VideoStream A data line 7
W4	SBData	serial bus data
W5	VCC_3.3	
W22	VCC_3.3	
W23	MDSF	Memory DSF line
W24	MRAS	Memory RAS line
W25	MCAS	Memory CAS line
W26	MClkE	Memory clock enable
Y1	VSAData_3	VideoStream A data line 3
Y2	VSAData_6	VideoStream A data line 6
Y3	VSAData_4	VideoStream A data line 4
Y4	PLLGND	PLL Power/Gnd pin

Table 8-1 Pinlist by Number

NO.	NAME	DESCRIPTION
Y5	VCC_3.3	
Y22	VCC_3.3	
Y23	MAddr_9	Memory address line 9
Y24	MAddr_10	Memory address line 10
Y25	MAddr_11	Memory address line 11
Y26	MWE	Memory write enable
AA1	RESERVED	No Connect
AA2	VSAdata_2	VideoStream A data line 2
AA3	VSAData_1	VideoStream A data line 1
AA4	VSAData_0	VideoStream A data line 0
AA5	GND	
AA22	GND	
AA23	MAddr_5	Memory address line 5
AA24	MAddr_6	Memory address line 6
AA25	MAddr_7	Memory address line 7
AA26	MAddr_8	Memory address line 8
AB1	PCIFIFOInDis	Delta control
AB2	PCIFIFOOutDis	Delta control
AB3	PCIRSTN	PCI reset
AB4	RESERVED	No Connect
AB5	GND	
AB6	GND	
AB7	VCC_3.3	
AB8	VCC_3.3	
AB9	GND	
AB10	GND	
AB11	VCC_3.3	
AB12	VCC_3.3	
AB13	GND	
AB14	GND	

Table 8-1 Pinlist by Number

NO.	NAME	DESCRIPTION
AB15	VCC_3.3	
AB16	VCC_3.3	
AB17	GND	
AB18	GND	
AB19	VCC_3.3	
AB20	VCC_3.3	
AB21	GND	
AB22	GND	
AB23	MemClkOut_0	Memory Clock Out 0
AB24	MAddr_2	Memory address line 2
AB25	MAddr_3	Memory address line 3
AB26	MAddr_4	Memory address line 4
AC1	PCIClkSel	33/66 MHz PCI Select
AC2	PCICLK	PCI clock
AC3	VDDQ_5	
AC4	VCC_2.5	
AC5	AGPSt_2	AGP status 2
AC6	AGPSBA_0	AGP Sideband Address 0
AC7	AGPSBA_3	AGP Sideband Address 3
AC8	AGPSBA_4	AGP Sideband Address 4
AC9	VDDQ_6	
AC10	PCIAD_29	PCI address/data line 29
AC11	PCIAD_26	PCI address/data line 26
AC12	PCIAD_23	PCI address/data line 23
AC13	PCIAD_21	PCI address/data line 21
AC14	PCIAD_20	PCI address/data line 20
AC15	PCICBEN_2	PCI byte enable 2
AC16	PCIFrameN	PCI frame signal
AC17	VDDQ_7	
AC18	PCIAD_15	PCI address/data line 15

Table 8-1 Pinlist by Number

NO.	NAME	DESCRIPTION
AC19	PCIAD_12	PCI address/data line 12
AC20	PCIAD_8	PCI address/data line 8
AC21	AGPADSTB0N	AGP AD 2X strobe
AC22	PCIAD_4	PCI address/data line 4
AC23	VCC_2.5	
AC24	MemClkOut_1	Memory clock out 1
AC25	MAddr_0	Memory address line 0
AC26	MAddr_1	Memory address line 1
AD1	PCIIntAN	PCI interrupt
AD2	GND_0	
AD3	VCC_2.5	
AD4	AGPSt_0	AGP status 0
AD5	AGPRbfN	AGP Read Data Buffer full
AD6	GND_1	
AD7	AGPSBSTB	AGP Sideband Address 2X strobe
AD8	AGPSBA_5	AGP Sideband Address 5
AD9	AGPSBA_7	AGP Sideband Address 7
AD10	GND_2	
AD11	PCIAD_25	PCI address/data line 25
AD12	PCIAD_24	PCI address/data line 24
AD13	PCIAD_22	PCI address/data line 22
AD14	VDDQ_3	
AD15	PCIAD_16	PCI address/data line 16
AD16	PCIDevSelN	PCI device select
AD17	PCIStopN	PCI stop
AD18	GND_4	
AD19	PCIAD_11	PCI address/data line 11
AD20	PCICBEN_0	PCI byte enable 0
AD21	PCIAD_7	PCI address/data line 7
AD22	VDDQ_8	

Table 8-1 Pinlist by Number

NO.	NAME	DESCRIPTION
AD23	AGPvREF	not connected
AD24	VCC_2.5	
AD25	MemClkOut_2	Memory Clock Out 2
AD26	MemClkOut_3	Memory Clock Out 3
AE1	VCC_2.5	
AE2	VCC_2.5	
AE3	PCIReqN	PCI request
AE4	AGPSt_1	AGP status 1
AE5	AGPPipeN	AGP Pipelined Address
AE6	AGPSBA_1	AGP Sideband Address 1
AE7	AGPSBSTEN	
AE8	AGPSBA_6	AGP Sideband Address 6
AE9	PCIAD_31	PCI address/data line 31
AE10	PCIAD_28	PCI address/data line 28
AE11	AGPADSTB1N	
AE12	PCICBEN_3	PCI byte enable 3
AE13	PCIAD_19	PCI address/data line 19
AE14	PCIAD_17	PCI address/data line 17
AE15	PCIIRdyN	PCI parity
AE16	PCITRdyN	PCI T ready
AE17	PCIPar	PCI ready
AE18	PCIAD_14	PCI address/data line 14
AE19	PCIAD_10	PCI address/data line 10
AE20	AGPADSTB0	AGP AD 2X strobe
AE21	PCIAD_6	PCI address/data line 6
AE22	PCIAD_3	PCI address/data line 3
AE23	PCIAD_1	PCI address/data line 1
AE24	RESERVED	
AE25	VCC_2.5	
AE26	VCC_2.5	

Table 8-1 Pinlist by Number

NO.	NAME	DESCRIPTION
AF1	VCC_2.5	
AF2	VCC_2.5	
AF3	PCIGntN	PCI grant signal
AF4	VDDQ_0	
AF5	WbfN	
AF6	AGPSBA_2	AGP Sideband Address 2
AF7	PINAGPTol0	
AF8	VDDQ_1	
AF9	PCIAD_30	PCI address/data line 30
AF10	PCIAD_27	PCI address/data line 27
AF11	AGPADSTB1	AGP AD 2X strobe
AF12	VDDQ_2	
AF13	PCIIdSel	PCI ID select
AF14	PCIAD_18	PCI address/data line 18
AF15	AGPtol1	V tolerant AGP I/Os
AF16	GND_3	
AF17	PCICBEN_1	PCI byte enable 1
AF18	PCIAD_13	PCI address/data line 13
AF19	PCIAD_9	PCI address/data line 9
AF20	VDDQ_4	
AF21	PCIAD_5	PCI address/data line 5
AF22	PCIAD_2	PCI address/data line 2
AF23	PCIAD_0	PCI address/data line 0
AF24	GND_5	
AF25	VCC_2.5	
AF26	VCC_2.5	

8.2 Pinlist by Name

The table below provides a brief description of each pin. It is organized alphabetically by pin name.

The pin type definitions used are:

- I/O: Input Signal
- GND: Ground
- VSS 3.3: Power at 3.3V
- VSS 2.5: Power at 2.5 Volts

Table 8-2 Pinlist by Name

NAME	NO.	DESCRIPTION
AGPADSTB0	AE20	AGP AD 2X strobe
AGPADSTB1	AF11	AGP AD 2X strobe
AGPADSTB0N	AC21	AGP AD 2x strobe
AGPADSTB1N	AE11	
AGPPipeN	AE5	AGP Pipelined address
AGPRbfN	AD5	AGP Read Data Buffer full
AGPSBA_0	AC6	AGP Sideband Address 0
AGPSBA_1	AE6	AGP Sideband Address 1
AGPSBA_2	AF6	AGP Sideband Address 2
AGPSBA_3	AC7	AGP Sideband Address 3
AGPSBA_4	AC8	AGP Sideband Address 4
AGPSBA_5	AD8	AGP Sideband Address 5
AGPSBA_6	AE8	AGP Sideband Address 6
AGPSBA_7	AD9	AGP Sideband Address 7
AGPSBSTB	AD7	AGP Sideband Address 2X strobe
AGPSBSTEN	AE7	
AGPSt_0	AD4	AGP status 0
AGPSt_1	AE4	AGP status 1
AGPSt_2	AC5	AGP status 2
AGPtol1	AF15	V tolerant AGP I/Os
AGPvREF	AD23	no connection
DacAGnd	R4	DAC Power/Gnd pin
DacAVDD	N1	Analog Video DAC
DacComp	P2	Compensation pin
GND	E5	
GND	E6	
GND	E9	
GND	E10	
GND	E13	
GND	E14	
GND	E17	
GND	E18	
GND	E21	
GND	E22	
GND	F5	

Table 8-2 Pinlist by Name

NAME	NO.	DESCRIPTION
GND	F22	
GND	J5	
GND	J22	
GND	K5	
GND	K22	
GND	L11	
GND	L12	
GND	L13	
GND	L14	
GND	L15	
GND	L16	
GND	M11	
GND	M12	
GND	M13	
GND	M14	
GND	M15	
GND	M16	
GND	N5	
GND	N11	
GND	N12	
GND	N13	
GND	N14	
GND	N15	
GND	N16	
GND	N22	
GND	P5	
GND	P11	
GND	P12	
GND	P13	
GND	P14	
GND	P15	
GND	P16	
GND	P22	
GND	R11	
GND	R12	
GND	R13	
GND	R14	
GND	R15	
GND	R16	
GND	T11	
GND	T12	
GND	T13	
GND	T14	
GND	T15	
GND	T16	
GND	U5	
GND	U22	
GND	V5	
GND	V22	
GND	AA5	
GND	AA22	
GND	AB5	

Table 8-2 Pinlist by Name

NAME	NO.	DESCRIPTION
GND	AB6	
GND	AB9	
GND	AB10	
GND	AB13	
GND	AB14	
GND	AB17	
GND	AB18	
GND	AB21	
GND	AB22	
GND_0	AD2	
GND_1	AD6	
GND_2	AD10	
GND_3	AF16	
GND_4	AD18	
GND_5	AF24	
MAddr_0	AC25	Memory address line 0
MAddr_1	AC26	Memory address line 1
MAddr_2	AB24	Memory address line 2
MAddr_3	AB25	Memory address line 3
MAddr_4	AB26	Memory address line 4
MAddr_5	AA23	Memory address line 5
MAddr_6	AA24	Memory address line 6
MAddr_7	AA25	Memory address line 7
MAddr_8	AA26	Memory address line 8
MAddr_9	Y23	Memory address line 9
MAddr_10	Y24	Memory address line 10
MAddr_11	Y25	Memory address line 11
MBank_0	V23	Memory bank select 0
MBank_1	V24	Memory bank select 1
MBank_2	U23	Memory bank select 2
MBank_3	J26	Memory bank select 3
MByte_0	U24	Memory byte select 0
MByte_1	U26	Memory byte select 1
MByte_2	V25	Memory byte select 2
MByte_3	U25	Memory byte select 3
MByte_4	G23	Memory byte select 4
MByte_5	H25	Memory byte select 5
MByte_6	H24	Memory byte select 6
MByte_7	G24	Memory byte select 7
MByte_8	C18	Memory byte select 8
MByte_9	B19	Memory byte select 9
MByte_10	C19	Memory byte select 10
MByte_11	B18	Memory byte select 11
MByte_12	D8	Memory byte select 12
MByte_13	C9	Memory byte select 13
MByte_14	D9	Memory byte select 14
MByte_15	B9	Memory byte select 15
MCAS	W25	Memory CAS line
MClkE	W26	Memory clock enable
MDat_0	L24	Memory data line 0
MDat_1	L25	Memory data line 1
MDat_2	L26	Memory data line 2

Table 8-2 Pinlist by Name

NAME	NO.	DESCRIPTION
MDat_3	L23	Memory data line 3
MDat_4	M26	Memory data line 4
MDat_5	M25	Memory data line 5
MDat_6	M24	Memory data line 6
MDat_7	M23	Memory data line 7
MDat_8	T24	Memory data line 8
MDat_9	T25	Memory data line 9
MDat_10	T26	Memory data line 10
MDat_11	T23	Memory data line 11
MDat_12	R26	Memory data line 12
MDat_13	R25	Memory data line 13
MDat_14	R24	Memory data line 14
MDat_15	R23	Memory data line 15
MDat_16	N23	Memory data line 16
MDat_17	N26	Memory data line 17
MDat_18	N25	Memory data line 18
MDat_19	N24	Memory data line 19
MDat_20	P23	Memory data line 20
MDat_21	P26	Memory data line 21
MDat_22	P25	Memory data line 22
MDat_23	P24	Memory data line 23
MDat_24	K23	Memory data line 24
MDat_25	K24	Memory data line 25
MDat_26	K25	Memory data line 26
MDat_27	K26	Memory data line 27
MDat_28	J25	Memory data line 28
MDat_29	J24	Memory data line 29
MDat_30	J23	Memory data line 30
MDat_31	H23	Memory data line 31
MDat_32	D21	Memory data line 32
MDat_33	C22	Memory data line 33
MDat_34	B22	Memory data line 34
MDat_35	A22	Memory data line 35
MDat_36	A23	Memory data line 36
MDat_37	B23	Memory data line 37
MDat_38	A24	Memory data line 38
MDat_39	B24	Memory data line 39
MDat_40	G25	Memory data line 40
MDat_41	G26	Memory data line 41
MDat_42	F26	Memory data line 42
MDat_43	F25	Memory data line 43
MDat_44	F24	Memory data line 44
MDat_45	F23	Memory data line 45
MDat_46	E24	Memory data line 46
MDat_47	E23	Memory data line 47
MDat_48	C23	Memory data line 48
MDat_49	C25	Memory data line 49
MDat_50	C26	Memory data line 50
MDat_51	D24	Memory data line 51
MDat_52	D25	Memory data line 52
MDat_53	D26	Memory data line 53
MDat_54	E26	Memory data line 54

Table 8-2 Pinlist by Name

NAME	NO.	DESCRIPTION
MDat_55	E25	Memory data line 55
MDat_56	D20	Memory data line 56
MDat_57	C21	Memory data line 57
MDat_58	B21	Memory data line 58
MDat_59	A21	Memory data line 59
MDat_60	A20	Memory data line 60
MDat_61	B20	Memory data line 61
MDat_62	C20	Memory data line 62
MDat_63	D19	Memory data line 63
MDat_64	D12	Memory data line 64
MDat_65	C12	Memory data line 65
MDat_66	B12	Memory data line 66
MDat_67	C13	Memory data line 67
MDat_68	D14	Memory data line 68
MDat_69	B13	Memory data line 69
MDat_70	A13	Memory data line 70
MDat_71	D13	Memory data line 71
MDat_72	D17	Memory data line 72
MDat_73	C17	Memory data line 73
MDat_74	B17	Memory data line 74
MDat_75	A18	Memory data line 75
MDat_76	A17	Memory data line 76
MDat_77	A16	Memory data line 77
MDat_78	B16	Memory data line 78
MDat_79	C16	Memory data line 79
MDat_80	C14	Memory data line 80
MDat_81	B14	Memory data line 81
MDat_82	A14	Memory data line 82
MDat_83	A15	Memory data line 83
MDat_84	D16	Memory data line 84
MDat_85	B15	Memory data line 85
MDat_86	C15	Memory data line 86
MDat_87	D15	Memory data line 87
MDat_88	C11	Memory data line 88
MDat_89	B11	Memory data line 89
MDat_90	A11	Memory data line 90
MDat_91	A12	Memory data line 91
MDat_92	D11	Memory data line 92
MDat_93	B10	Memory data line 93
MDat_94	C10	Memory data line 94
MDat_95	D10	Memory data line 95
MDat_96	C4	Memory data line 96
MDat_97	B4	Memory data line 97
MDat_98	A4	Memory data line 98
MDat_99	B3	Memory data line 99
MDat_100	D2	Memory data line 100
MDat_101	E2	Memory data line 101
MDat_102	E3	Memory data line 102
MDat_103	F3	Memory data line 103
MDat_104	A9	Memory data line 104
MDat_105	D7	Memory data line 105
MDat_106	C8	Memory data line 106

Table 8-2 Pinlist by Name

NAME	NO.	DESCRIPTION
MDat_107	B8	Memory data line 107
MDat_108	B7	Memory data line 108
MDat_109	C7	Memory data line 109
MDat_110	D6	Memory data line 110
MDat_111	A8	Memory data line 111
MDat_112	A6	Memory data line 112
MDat_113	A5	Memory data line 113
MDat_114	C5	Memory data line 114
MDat_115	B5	Memory data line 115
MDat_116	B6	Memory data line 116
MDat_117	C6	Memory data line 117
MDat_118	D5	Memory data line 118
MDat_119	A7	Memory data line 119
MDat_120	F1	Memory data line 120
MDat_121	E1	Memory data line 121
MDat_122	D1	Memory data line 122
MDat_123	C1	Memory data line 123
MDat_124	A3	Memory data line 124
MDat_125	C2	Memory data line 125
MDat_126	D3	Memory data line 126
MDat_127	E4	Memory data line 127
MDSF	W23	Memory DSF line
MemClkOut_0	AB23	Memory Clock Out 0
MemClkOut_1	AC24	Memory Clock Out 1
MemClkOut_2	AD25	Memory Clock Out 2
MemClkOut_3	AD26	Memory Clock Out 3
MemClkRet_0	V26	Memory Clock Return 0
MemClkRet_1	H26	Memory Clock Return 1
MemClkRet_2	A19	Memory Clock Return 2
MemClkRet_3	A10	Memory Clock Return 3
MRAS	W24	Memory RAS line
MWE	Y26	Memory write enable
PCIAD_0	AF23	PCI address/data line 0
PCIAD_1	AE23	PCI address/data line 1
PCIAD_10	AE19	PCI address/data line 10
PCIAD_11	AD19	PCI address/data line 11
PCIAD_12	AC19	PCI address/data line 12
PCIAD_13	AF18	PCI address/data line 13
PCIAD_14	AE18	PCI address/data line 14
PCIAD_15	AC18	PCI address/data line 15
PCIAD_16	AD15	PCI address/data line 16
PCIAD_17	AE14	PCI address/data line 17
PCIAD_18	AF14	PCI address/data line 18
PCIAD_19	AE13	PCI address/data line 19
PCIAD_2	AF22	PCI address/data line 2
PCIAD_20	AC14	PCI address/data line 20
PCIAD_21	AC13	PCI address/data line 21
PCIAD_22	AD13	PCI address/data line 22
PCIAD_23	AC12	PCI address/data line 23
PCIAD_24	AD12	PCI address/data line 24
PCIAD_25	AD11	PCI address/data line 25
PCIAD_26	AC11	PCI address/data line 26

Table 8-2 Pinlist by Name

NAME	NO.	DESCRIPTION
PCIAD_27	AF10	PCI address/data line 27
PCIAD_28	AE10	PCI address/data line 28
PCIAD_29	AC10	PCI address/data line 29
PCIAD_3	AE22	PCI address/data line 3
PCIAD_30	AF9	PCI address/data line 30
PCIAD_31	AE9	PCI address/data line 31
PCIAD_4	AC22	PCI address/data line 4
PCIAD_5	AF21	PCI address/data line 5
PCIAD_6	AE21	PCI address/data line 6
PCIAD_7	AD21	PCI address/data line 7
PCIAD_8	AC20	PCI address/data line 8
PCIAD_9	AF19	PCI address/data line 9
PCICBEN_0	AD20	PCI byte enable 0
PCICBEN_1	AF17	PCI byte enable 1
PCICBEN_2	AC15	PCI byte enable 2
PCICBEN_3	AE12	PCI byte enable 3
PCICLK	AC2	PCI clock
PCIClkSel	AC1	33/66 MHz PCI select
PCIDevSelN	AD16	PCI device select
PCIFIFOInDis	AB1	Delta control
PCIFIFOOutDis	AB2	Delta control
PCIFrameN	AC16	PCI frame signal
PCIGntN	AF3	PCI grant signal
PCIIdSel	AF13	PCI ID select
PCIIntAN	AD1	PCI interrupt
PCIIRdyN	AE15	PCI parity
PCIPar	AE17	PCI ready
PCIReqN	AE3	PCI request
PCIRSTN	AB3	PCI reset
PCIStopN	AD17	PCI stop
PCITRdyN	AE16	PCI T ready
PINAGPTol0	AF7	
PLLDISABLE	T3	PLL Disable
PLLPower	W1	PLL Power/Gnd pin
PLLGND	Y4	PLL Power/Gnd pin
RenderSyncN	D18	Multirasterizer i/o sync pin
RESERVED	AE24	
RESERVED	AA1	No Connect
RESERVED	AB4	No Connect
ROMSelN	U2	ROM select signal
ROMWeN	U1	ROM write wnable
SBClk	V1	serial bus clock
SBData	W4	serial bus data
SPARE	V2	
SPARE	J4	
TestMode	R1	Test Mode control
TestSel_0_	M1	Test Mode select 0
TestSel_1	R3	Test Mode select
TestSel_2_	M2	Test Mode select 1
VCC_2.5	D4	
VCC_2.5	A1	
VCC_2.5	B1	

Table 8-2 Pinlist by Name

NAME	NO.	DESCRIPTION
VCC_2.5	AE1	
VCC_2.5	AE2	
VCC_2.5	AD3	
VCC_2.5	AC4	
VCC_2.5	AF1	
VCC_2.5	AF2	
VCC_2.5	AF25	
VCC_2.5	AE25	
VCC_2.5	AD24	
VCC_2.5	AC23	
VCC_2.5	AF26	
VCC_2.5	AE26	
VCC_2.5	B26	
VCC_2.5	B25	
VCC_2.5	C24	
VCC_2.5	D23	
VCC_2.5	A26	
VCC_2.5	A25	
VCC_2.5	B2	
VCC_2.5	A2	
VCC_2.5	C3	
VCC_3.3	E7	
VCC_3.3	E8	
VCC_3.3	E11	
VCC_3.3	E12	
VCC_3.3	E15	
VCC_3.3	E16	
VCC_3.3	E19	
VCC_3.3	E20	
VCC_3.3	G5	
VCC_3.3	G22	
VCC_3.3	H5	
VCC_3.3	H22	
VCC_3.3	L5	
VCC_3.3	L22	
VCC_3.3	M5	
VCC_3.3	M22	
VCC_3.3	R5	
VCC_3.3	R22	
VCC_3.3	T5	
VCC_3.3	T22	
VCC_3.3	W5	
VCC_3.3	W22	
VCC_3.3	Y5	
VCC_3.3	Y22	
VCC_3.3	AB7	
VCC_3.3	AB8	
VCC_3.3	AB11	
VCC_3.3	AB12	
VCC_3.3	AB15	
VCC_3.3	AB16	
VCC_3.3	AB19	

Table 8-2 Pinlist by Name

NAME	NO.	DESCRIPTION
VCC_3.3	AB20	
VDDQ_0	AF4	
VDDQ_1	AF8	
VDDQ_2	AF12	
VDDQ_3	AD14	
VDDQ_4	AF20	
VDDQ_5	AC3	
VDDQ_6	AC9	
VDDQ_7	AC17	
VDDQ_8	AD22	
vidBlue	P3	Analog blue signal
VidDDCClk	U4	Clock line for DDC
VidDDCData	U3	Data line for DDC
VideoExtCtrl	D22	Video external control
vidGreen	N3	Analog green signal
VidHSync	R2	Horizontal sync
vidRed	N2	Analog red signal
vidResRef	P4	Reference resistor
VidRightEye	N4	Right signal for stereo
VidVRef	P1	Voltage reference
VidVsync	T2	Vertical sync
VSAClk	V3	VideoStream A clock
VSAData_0	AA4	VideoStream A data line 0
VSAData_1	AA3	VideoStream A data line 1
VSAdata_2	AA2	VideoStream A data line 2
VSAData_3	Y1	VideoStream A data line 3
VSAData_4	Y3	VideoStream A data line 4
VSAData_5	W2	VideoStream A data line 5
VSAData_6	Y2	VideoStream A data line 6
VSAData_7	W3	VideoStream A data line 7
VSAResetN	V4	Video Stream reset
VSBClk	H4	VideoStream B clock
VSBClkOut	J3	Video Stream B clock out
VSBData_0	F4	VideoStream B data line 0
VSBData_1	G1	VideoStream B data line 1
VSBData_2	G2	VideoStream B data line 2
VSBData_3	G3	VideoStream B data line 3
VSBData_4	G4	VideoStream B data line 4
VSBData_5	H1	VideoStream B data line 5
VSBData_6	H2	VideoStream B data line 6
VSBData_7	H3	VideoStream B data line 7
VSBResetN	F2	Video Stream B Reset Out
VSCtl_0	K1	VideoStreams Control line 0
VSCtl_1	K2	VideoStreams Control line 1
VSCtl_2	L4	VideoStreams Control line 2
VSCtl_3	L1	VideoStreams Control line 3
VSCtl_4	L2	VideoStreams Control line 4
VSCtl_5	L3	VideoStreams Control line 5
VSCtl_6	M3	VideoStreams Control line 6
VSCtl_7	M4	VideoStreams Control line 7
VSGPChipSelectN	K3	VS GP bus chip select
VSGPDataAckN	K4	VS GP bus data ack

Table 8-2 Pinlist by Name

NAME	NO.	DESCRIPTION
VSGPDataStrob eN	J1	VS GP bus data strobe
VSGPReadWrit eN	J2	VS GP bus read/write signal
WbfN	AF5	
Xtal1	T1	Crystal i/p 1
Xtal2	T4	Crystal i/p 2

9

Memory System

The PERMEDIA memory system is intended for use with Synchronous Dynamic Memories. The memories can be SGRAM or SDRAM devices. The width of the memory interface is 128 bits, but can be configured to 64 bits. Control lines are provided for 4 blocks of memories, these are Select (3 – 0). Four ClockOut and ClockReturn signals are also provided, these are to assist in de-skewing the return data and reducing the load on each clock line. The Clock lines should be wired as illustrated in Figure 9.1. The memory system has one set of primary control signals which are common to all blocks, these are Data, Address, RAS, CAS, WriteEnable (WE), DSF, ClkEnable and Byte enables (DQM). A typical organization is shown below.

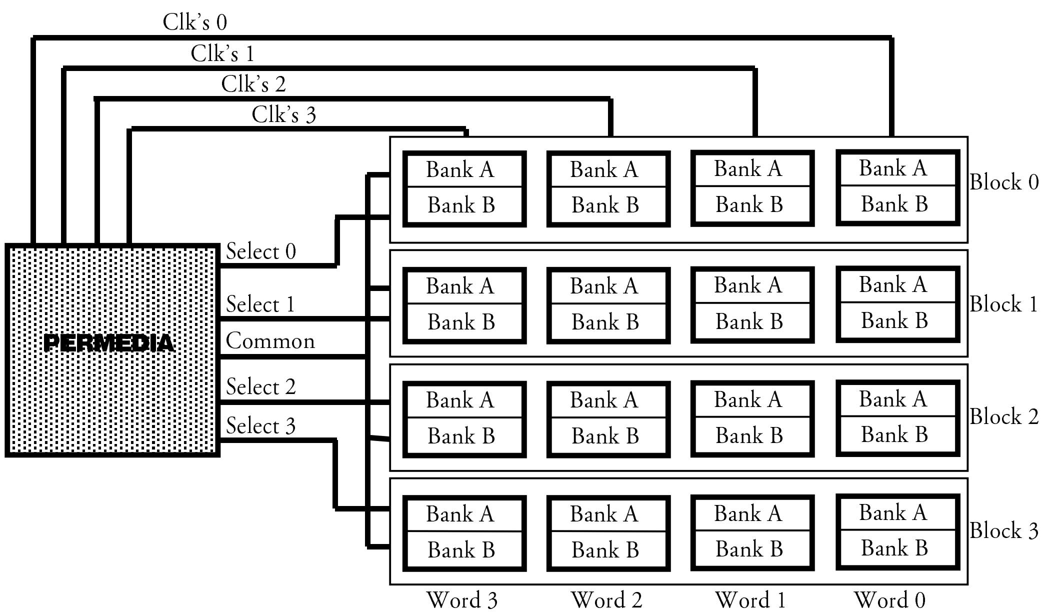


Figure 9.1 **Organization of memory devices**

The diagram shows a 16-megabyte memory array, constructed from 16, 8-Megabit memories, arranged into 4 blocks. The devices used are 32 bit wide with 2 banks, where each bank has 512 rows and 256 columns.

9.1 System Parameters

The Memory System employs a rich set of registers, which allow for a diverse range of memory configurations. The various timing parameters used to control synchronous memories can be adjusted to allow for optimum performance depending on memory type, speed grade and the PERMEDIA system clock frequency (MClk). Memory functionally can be enabled depending on the type fitted. Full addressing control is available so that virtually any memory configuration can be fitted.

The following parameters are used to control accesses to the memory. These values fall into three categories

- Addressing
- Functionality and Optimizations
- Timing and Mode

9.1.1 Addressing

9.1.1.1 ColumnAddress

This parameter defines the number of address bits required to generate the column addresses for the memory devices fitted. This parameter is normally quoted in the memory device data sheet.

For example CA7~CA0 therefore the Column Address parameter would be 8

9.1.1.2 RowAddress

This parameter defines the number of address bits required to generate the row addresses for the memory device fitted. This parameter is normally quoted in the memory device data sheet.

For example RA8~RA0 therefore the Row Address parameter would be 9

9.1.1.3 BankAddress

This parameter defines the number of address bits required to generate the bank addresses for the memory device fitted. This parameter is normally quoted in the memory device data sheet.

For example A9(BA) therefore the Bank Address parameter would be 1

9.1.1.4 ChipSelect

This parameter defines the number of address bits needed to select all the blocks of memory devices fitted to the PERMEDIA device.

For 1 Block of memories	Chip Select = 1
For 2 Blocks of memories	Chip Select = 1
For 3 Blocks of memories	Chip Select = 2
For 4 Blocks of memories	Chip Select = 2

9.1.1.5 PageSize

This parameter defines the address range for a memory page of the memory array fitted. The value can be calculated as (column address bits of device – 5). The PageSize parameter modified if either Interleave (0) or Halfwidth (9.1.1.9) are set. PageSize can be calculated as ((column address bits) – 5) + Interleave – Halfwidth.

9.1.1.6 RegionSize

This parameter defines the addressing range for each of the four page-detectors implemented in the memory controller. The minimum region a page-detector can be assigned to is one internal bank, the maximum is all of the memory fitted. There are some memory configurations where not all the page-detectors can be deployed. An example of this is when three blocks of memory devices are used. The value can be calculated as

Where

$$\mathrm{Log2}\left(\frac{\mathrm{TotalMemory}}{\mathrm{BytesperMemWidth} \times \mathrm{RegionsUsed}}\right) - 5$$

TotalMemory = The total size of memory fitted in megabytes

Bytes per Memory Width = 16 (128 / 8)

Regions Used = (if total number of Banks (Blocks fitted x Internal Banks) > 4
then Blocks Fitted
else Total Banks)

As an example the memory configuration in Figure 9-1 is constructed from sixteen 8-megabit devices each with two internal banks

TotalMemory = 16777216 (16-megabytes)

Bytes per Memory Width = 16

Regions Used = (Blocks fitted = 4) x (Internal Banks = 2) = 8
= 8 > 4

$$= 4$$

$$\text{Log2}\left(\frac{16777216}{16 \times 4}\right) - 5$$

RegionSize = 13

9.1.1.7 CombineBanks

This flag should be set, when the total number of banks fitted is greater than 4. The total number of banks can be determined by multiplying the number of internal banks of the device by the number of device blocks fitted. In the example shown in Figure 9-1, there are 4 device blocks fitted (Blocks 0 to 3), each device has 2 internal banks (Banks A and B), so the total number of banks is 8, therefore CombineBanks should be set.

9.1.1.8 InterLeave

This flag when set doubles the page size of the memory array. This is accomplished by combining two blocks of memory and operating them as one. Both blocks are PRECHARGED and ACTIVATED together, and any command sequences issued that cross from one block to the other, do so without incurring a page brake. From the example configuration detailed in Figure 9-1, Block 1 would interleave with Block 0, and Block 3 with Block 2. When this flag is set the value loaded into the PageSize parameter (9.1.1.5) should be increased by one. As the Blocks are now operating in pairs the total number of banks fitted is halved. This may have a bearing on the CombineBanks flag (9.1.1.7).

9.1.1.9 HalfWidth

This flag should be set only when the memory buffer fitted is 64 bits wide. When set, this flag has an impact on the PageSize register, (section 9.1.1.5).

9.1.2 Controlling larger memory devices

Permedia3 can drive 64MBx32bit memory devices as follows:

- Tie the CS of the memories Low
- Wire chip Address lines A10 to A0 to memory address lines A10 to A0,
- Wire chip Address line A11 to memory BA0
- Wire chip BankSelect0 to memory BA1

LocalMemCaps register configuration:

Load the register with the value 0x30E311B8. This sets the following parameters:

- Cas address bits 8
- Ras address bits 11
- Bank address bits 1
- ChipSelect bits 1
- Region Size 14

This tactic 'tricks' the Memory Controller into operating as if 2 blocks of twin bank devices are fitted. This approach is reliable and used on a number of **3D***Labs* board products.

9.1.3 Functionality and Optimizations

9.1.3.1 NoPrechargeOpt

This flag when set will disable the back to back READ - PRECHARGE optimization, inserting clocks to the value of the CAS Latency between the commands. If the memory devices fitted are capable of executing a READ command directly followed by a PRECHARGE command, this flag should be left clear for optimal performance. The bit setting cannot be read back directly and should be set or reset when in doubt.

9.1.3.2 SpecialModeOpt

This flag when set enables the memory controller to issue a Special Mode Register Set (SMRS) command, without regard to the current state of the internal banks of the SGRAM. Some memory devices require all internal banks to be in the same state before an SMRS command is issued. For these devices, ensure that the flag is cleared. The memory controller will issue a PRECHARGE command to the devices to ensure all internal banks are in the IDLE mode before issuing the SMRS command. If the memory devices fitted are capable of this function, optimally this flag should be set.

9.1.3.3 TwoColorBlockFill

This flag when set allows the memory controller to utilize the 2 internal Color Registers that some SGRAM devices are equipped with. If the memory devices fitted only have 1 Color Register, this flag should be cleared. When this flag is cleared the memory controller will fully emulate the two color fill operations.

9.1.3.4 NoWriteMask

This flag when set disables the memory controller from using the internal MASK Register of an SGRAM. This flag must be set if SDRAMS are fitted. When this flag is set, the memory controller will emulate the write mask operations. This is only a partial emulation using the byte enables so bit precision is not achieved.

9.1.3.5 NoBlockFill

This flag when set disables the memory controller from issuing a Block Fill command to the memories. This flag must be set if SDRAMS are fitted. When this flag is set the memory controller will fully emulate the block fill operations.

9.1.3.6 NoLookAhead

This flag when set disables the memory controller from issuing command to one bank of memory, whilst another bank is in the process of PRECHARGHING. Nominally for performance, this flag should be left cleared.

9.1.4 Timing and Mode

9.1.4.1 TurnOn (Block to Block Read Delay)

This parameter defines the number of MClk cycles that need to be inserted between issuing a READ command to one block of memory devices to a READ of another Block. (Block to Block Read Delay). Two parameters from the memory device data sheet must be used to determine what value TurnOn must be set to. The timing parameter tHZ defines the tri-state time and the parameter tLZ defines the drive time of the device. If tLZ is greater than tHZ, then this parameter can safely be set to zero.

9.1.4.2 TurnOff (Read to Write Turn around)

This parameter defines the number of MClk cycles that need to be inserted between issuing a READ and a WRITE command (Read – Write turn around). This parameter is defined in the memory device data sheet, usually as tHZ.

9.1.4.3 RegisterLoad (RL)

This parameter defines the number of MClk cycles that need to be inserted between issuing a SMRS and another command. This parameter is usually detailed in the memory device data sheet as tRSC. If tRSC is quoted including the SMRS cycle, then RegisterLoad should be calculated as tRSC (in MClk cycles) – 1.

9.1.4.4 BlockWrite (BW)

This parameter defines the number of MClk cycles that need to be inserted between issuing a BLOCK WRITE and another command. This parameter is usually detailed in the memory device data sheet as tBWC. If tBWC is quoted including the SMRS cycle, then BlockWrite should be calculated as tBWC (in MClk cycles) – 1.

9.1.4.5 ActivateToCommand (ATC)

This parameter defines the number of MClk cycles that need to be inserted between issuing an ACTIVATE and a command. This parameter is usually detailed in the memory device data sheet as tRCD. If tRCD is quoted including the ACTIVATE cycle, then ActivateToCommand should be calculated as tRCD (in MClk cycles) – 1.

9.1.4.6 PrechargeToActivate (PTA)

This parameter defines the number of MClk cycles that need to be inserted between issuing a PRECHARGE and an ACTIVATE command. This parameter is usually detailed in the memory device data sheet as tRP. If tRP is quoted including the PRECHARGE cycle, then PreChargeToActivate should be calculated as tRP (in MClk cycles) – 1.

9.1.4.7 BlockWriteToPrecharge (BTP)

This parameter defines the number of MClk cycles that need to be inserted between issuing a BLOCKWRITE and a PRECHARGE command. This parameter is usually detailed in the memory device data sheet as tBPL (tBWR). If tBPL is quoted including the BLOCKWRITE cycle, then BlockWriteToPrecharge should be calculated as tBPL (in MClk cycles) – 1.

9.1.4.8 WriteToPrecharge (WTP)

This parameter defines the number of MClk cycles that need to be inserted between issuing a WRITE and a PRECHARGE command. This parameter is usually detailed in the memory device data sheet as tRDL (tWR). If tRDL is quoted including the WRITE cycle, then WriteToPrecharge should be calculated as tRDL (in MClk cycles) – 1.

9.1.4.9 ActivateToPrecharge (ATP)

This parameter defines the number of MClk cycles that need to be inserted between issuing an ACTIVATE and a PRECHARGE command. This parameter is usually detailed in the memory device data sheet as tRAS. If tRAS is quoted including the ACTIVATE cycle, then ActivateToPrecharge should be calculated as tRAS (in MClk cycles) – 1.

9.1.4.10 RefreshCycle (RC)

This parameter defines the number of MClk cycles that need to be inserted between issuing and REFRESH and an ACTIVATE command. This parameter is usually detailed in the memory device data sheet as tRC. If tRC is quoted including the REFRESH command cycle, then RefreshCycle should be calculated as tRC (in MClk cycles) – 1.

9.1.4.11 CasLatency (CL)

This parameter determines the CAS latency expected by the memory controller. The CasLatency parameter can be loaded directly with the appropriate value from the memory device data sheet. For example, if a CAS latency of 2 is required then the CasLatency parameter should be set to 2.

9.1.4.12 Mode

This parameter defines the value of the Mode Register loaded into the SGRAM at the end of the boot sequence (see data sheet). Items to note: Burst type should be sequential, burst length should be set to one and CAS latency should be consistent with the CASLatency parameter. For devices that have a Color Register field, this should be consistent with the TwoColorBlockFill flag. All other bits in the Mode field should be set low.

9.1.4.13 RefreshEnable

This flag should be set for Refresh commands to be issued by the memory controller.

9.1.4.14 RefreshCount

This parameter defines the period between AUTO-REFRESH commands being issued to the synchronous memories. The count is in ((MClks/32) + 16) i.e. if RefreshCount = 1, the synchronous memories will be refreshed every 48 MClk cycles. For the required refresh rate, see the synchronous memory data sheet.

9.2 Example Parameter Values

The following device types and values are given as examples and should not be taken as recommendations.

9.2.1 100MHz/Samsung KM4132G271A-10 SGRAM /total SGRAM 12MB

For a PERMEDIA device running with an MClk of 100MHz, fitted with 12, 8Mbit, 2x512x256x32 SGRAMS arranged into 3 Blocks.

Table 9.1 100MHz/Samsung KM4132G271A-10 SGRAM /total SGRAM 12MB

Addressing Parameters	Value (binary)	Comment
ColumnAddress	1000	8
RowAddress	1001	9
BankAddress	0001	1 (2 Banks A/B)
ChipSelect	0010	2 (3 Blocks)
PageSize	0011	3 (256)
RegionSize	1101	13 (4 MB)
CombineBanks	1	6 = 3 Blocks x 2Banks
Interleave	0	Optional
HalfWidth	0	Unavailable 128 bit

Functionality Parameters	Value (binary)	Comment
NoPrechargeOpt	0	Preferred
SpecialModeOpt	1	Preferred
TwoColorBlockFill	0	Only 1 Color Register
NoWriteMask	0	Preferred
NoBlockFill	0	Preferred
NoLookAhead	0	Preferred

Timing and Mode Parameters	Value (binary)	Comment
TurnOn (Block to Block Read)	01	Tshz 7ns
TurnOff (Read to Write)	01	Tshz 7ns
RegisterLoad (RL)	00	New command next Clk
BlockWrite (BW)	01	Tbwc 20ns, 2Clk -1
ActivateToCommand (ATC)	001	Trcd 20ns, 2Clk –1
PrechargeToActivate (PTA)	010	Trp 26ns, 3Clk -1
BlockWriteToPrecharge (BTP)	001	Tbpl 20ns, 2Clk -1
WriteToPrecharge (WTP)	000	Trdl 1Clk -1
ActivateToPrecharge (ATP)	0100	Tras 50ns, 5Clk –1
RefreshCycle (RC)	0111	Trc 80ns 8Clk -1
CasLatency (CL)	011	CasLatency 3
Mode	0000110000	CL3
RefreshEnable	1	
RefreshCount	0110000	64432 Refresh c/s

9.2.2 125MHz[1]/SIEMENS HYB39S16320-7 SGRAM /total SGRAM 16MB

For a PERMEDIA device running with an MClk of 125MHz, fitted with 8, 16Mbit, 2x1024x256x32 SGRAMS arranged into 2 Blocks.

Table 9.2 125MHz/SIEMENS HYB39S16320-7 SGRAM /total SGRAM 16MB

Addressing Parameters	Value (binary)	Comment
ColumnAddress	1000	8
RowAddress	1010	10
BankAddress	0001	1 (2 Banks A/B)
ChipSelect	0001	1 (2 Blocks)
PageSize	0011	3 (256)
RegionSize	1101	13 (4 MB)
CombineBanks	0	4 = 2 Blocks x 2Banks
Interleave	0	Optional
HalfWidth	0	Unavailable 128 bit

Functionality Parameters	Value (binary)	Comment
NoPrechargeOpt	0	Preferred
SpecialModeOpt	1	Preferred
TwoColorBlockFill	1	Preferred
NoWriteMask	0	Preferred
NoBlockFill	0	Preferred
NoLookAhead	0	Preferred

Timing and Mode Parameters	Value (binary)	Comment
TurnOn (Block to Block Read)	01	Thz 8ns
TurnOff (Read to Write)	01	Thz 8ns
RegisterLoad (RL)	01	Trsc 2Clk -1
BlockWrite (BW)	01	Tbwc 14ns, 2Clk -1
ActivateToCommand (ATC)	010	Trcd 21ns, 3Clk –1
PrechargeToActivate (PTA)	010	Trp 21ns, 3Clk -1
BlockWriteToPrecharge (BTP)	001	Tbwr 14ns, 2Clk -1
WriteToPrecharge (WTP)	000	Trdl 1Clk -1
ActivateToPrecharge (ATP)	0110	Tras 49ns, 7Clk –1
RefreshCycle (RC)	1001	Trc 70ns 9Clk -1
CasLatency (CL)	010	CasLatency 2
Mode	0001100000	2 Color Reg + CL2
RefreshEnable	1	
RefreshCount	0111100	64566 Refresh c/s

[1] Please note: 125MHz MClk is provisional.

9.2.3 125MHz2/ MICRON MT48LC1M16A1-8A SDRAM /total SDRAM 16MB

For a PERMEDIA device running with an MClk of 125MHz, fitted with 8, 16Mbit, 2x2048x256x16 SDRAMS arranged into 1 Block.

Table 9.3 125MHz/MICRON MT48LC1M16A1-8A SDRAM /total SDRAM 16MB

Addressing Parameters	Value (binary)	Comment
ColumnAddress	1000	8
RowAddress	1011	11
BankAddress	0001	1 A/B
ChipSelect	0001	1 (1 Blocks)
PageSize	0011	3 (256)
RegionSize	1101	14 (8 MB)
CombineBanks	0	2 = 1 Block x 2Banks
Interleave	0	Unavailable only 1 block
HalfWidth	0	Unavailable 128 bit

Functionality Parameters	Value (binary)	Comment
NoPrechargeOpt	0	Preferred
SpecialModeOpt	1	Preferred
TwoColorBlockFill	0	Unavailable
NoWriteMask	1	Unavailable
NoBlockFill	1	Unavailable
NoLookAhead	0	Preferred

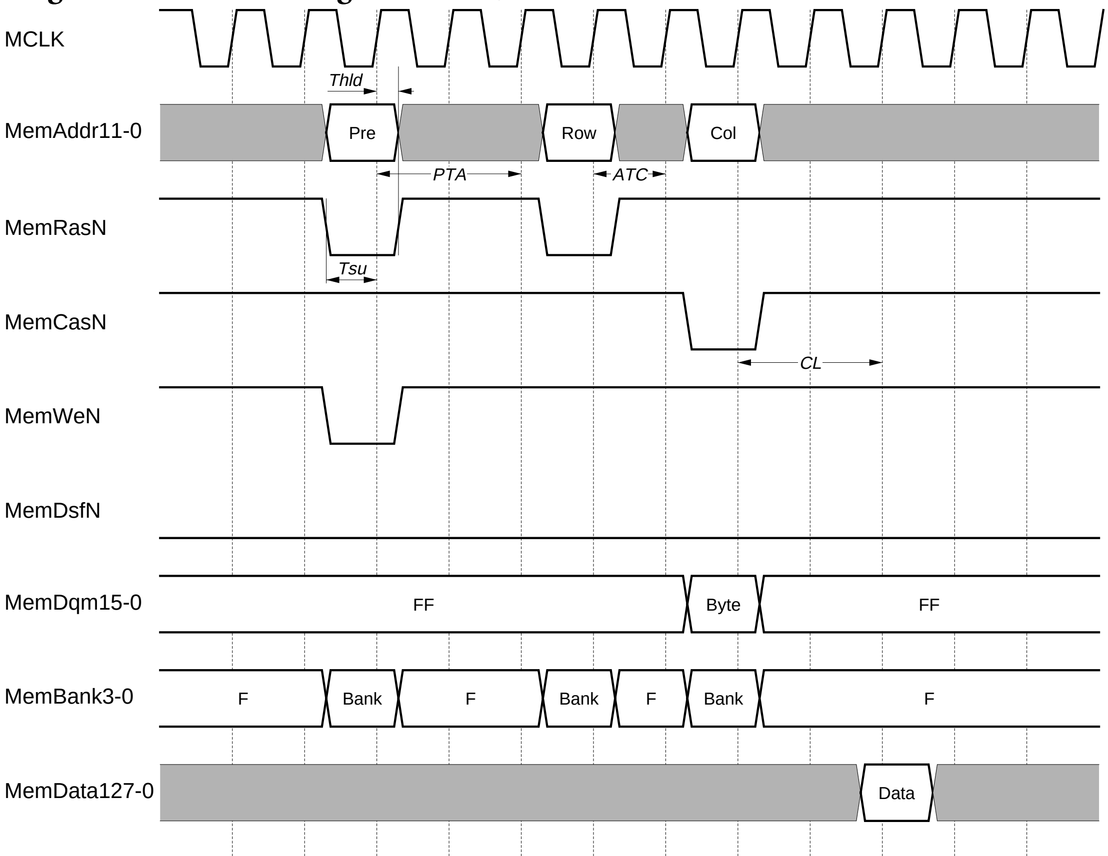
Timing and Mode Parameters	Value (binary)	Comment
TurnOn (Block to Block Read)	01	Thz 7ns
TurnOff (Read to Write)	01	Thz 7ns
RegisterLoad (RL)	01	Tmrd 2Clk -1
BlockWrite (BW)	00	NA
ActivateToCommand (ATC)	011	Trcd 30ns, 4Clk –1
PrechargeToActivate (PTA)	011	Trp 30ns, 4Clk -1
BlockWriteToPrecharge (BTP)	000	NA
WriteToPrecharge (WTP)	000	Twr 1Clk -1
ActivateToPrecharge (ATP)	0110	Tras 50ns, 7Clk –1
RefreshCycle (RC)	1001	Trc 80ns 10Clk -1
CasLatency (CL)	010	CasLatency 2
Mode	0000100000	CL2
RefreshEnable	1	
RefreshCount	0111100	64566 Refresh c/s

2 Please note: 125MHz MClk is provisional.

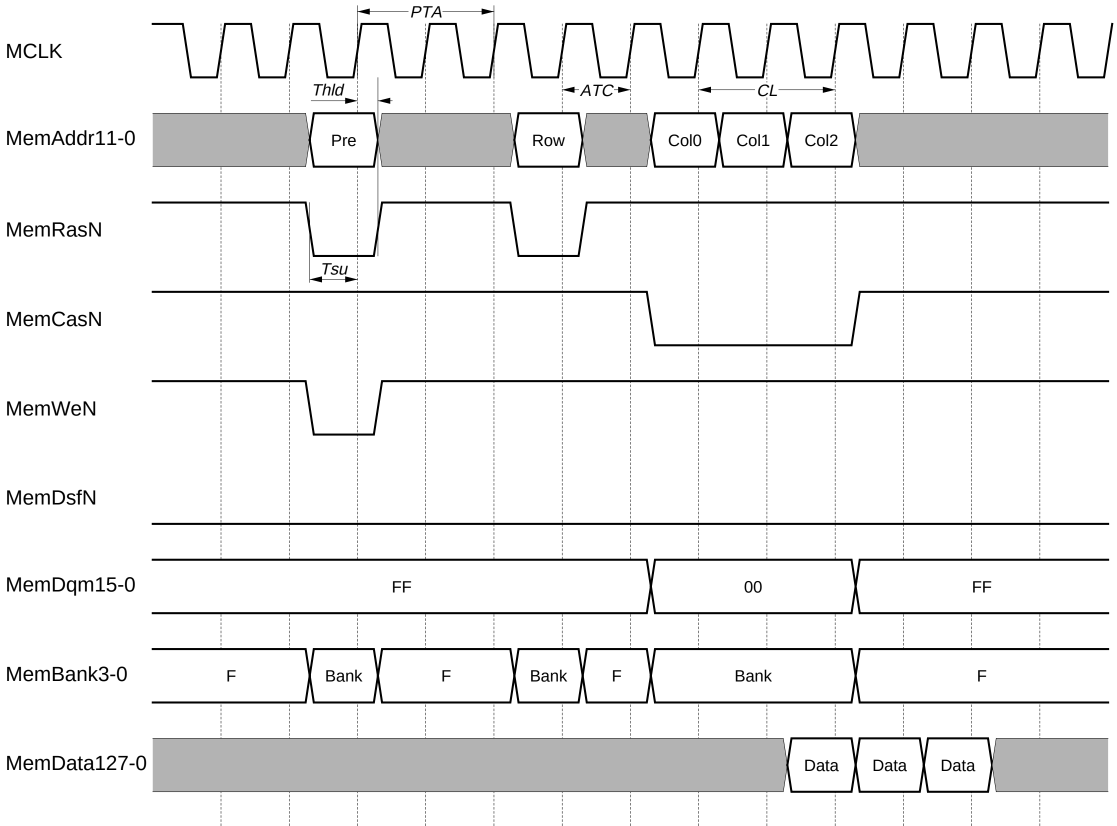
Timing Diagrams

The following timing diagrams show specific operations of the memory controller.

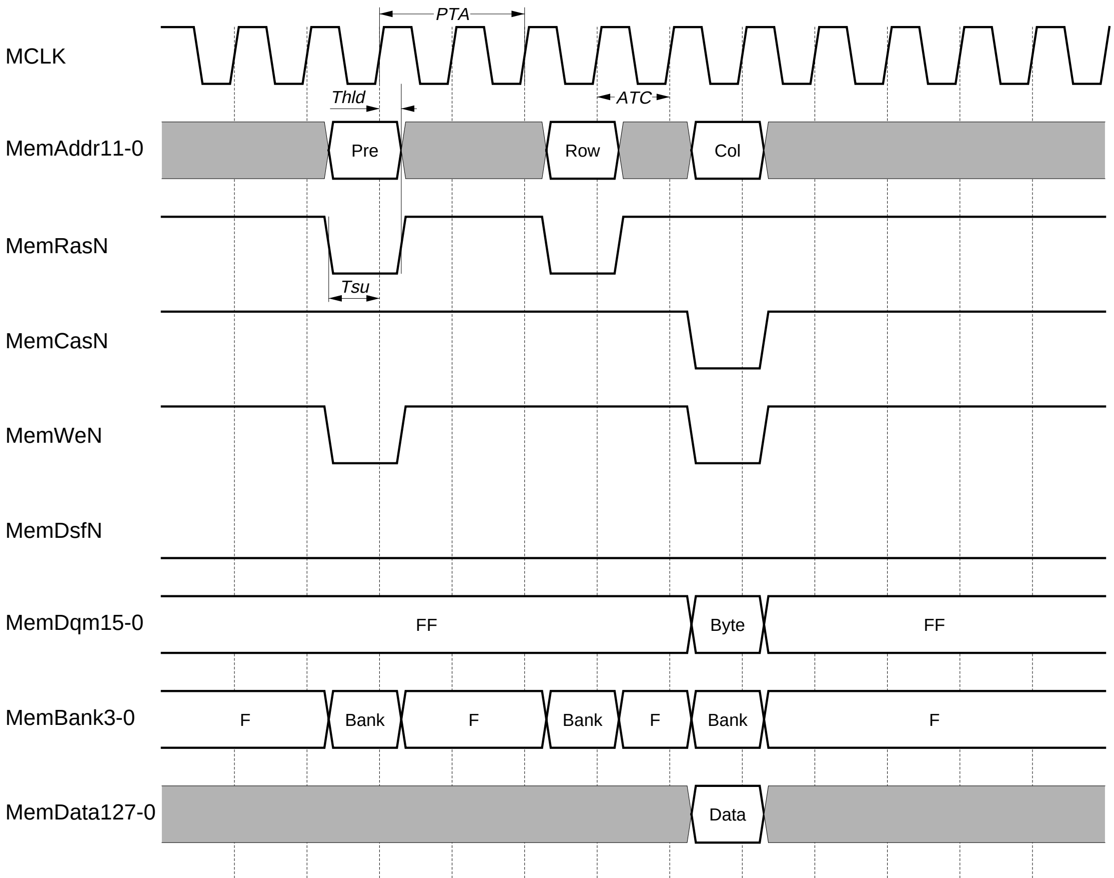
Single Read with Precharge @ CL = 2, PTA = 2 ATC = 1



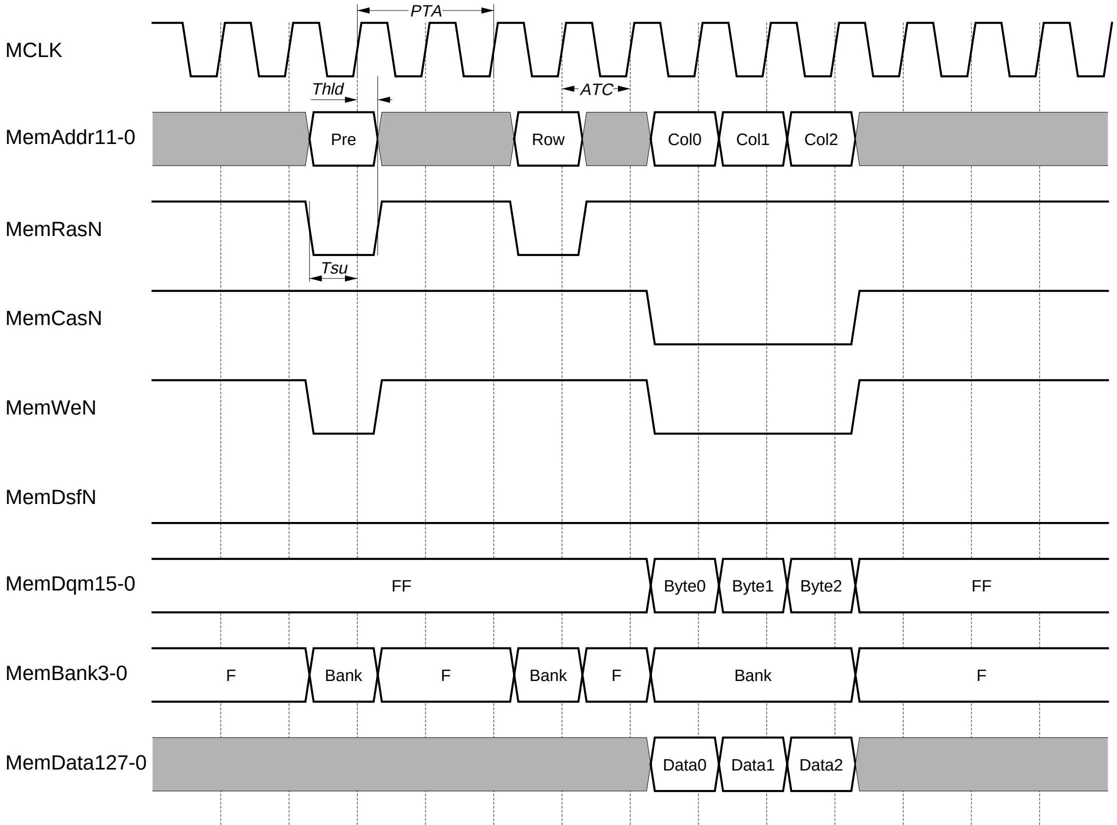
Multiple Reads to Same Bank @ CL = 2, PTA = 2 , ATC = 1



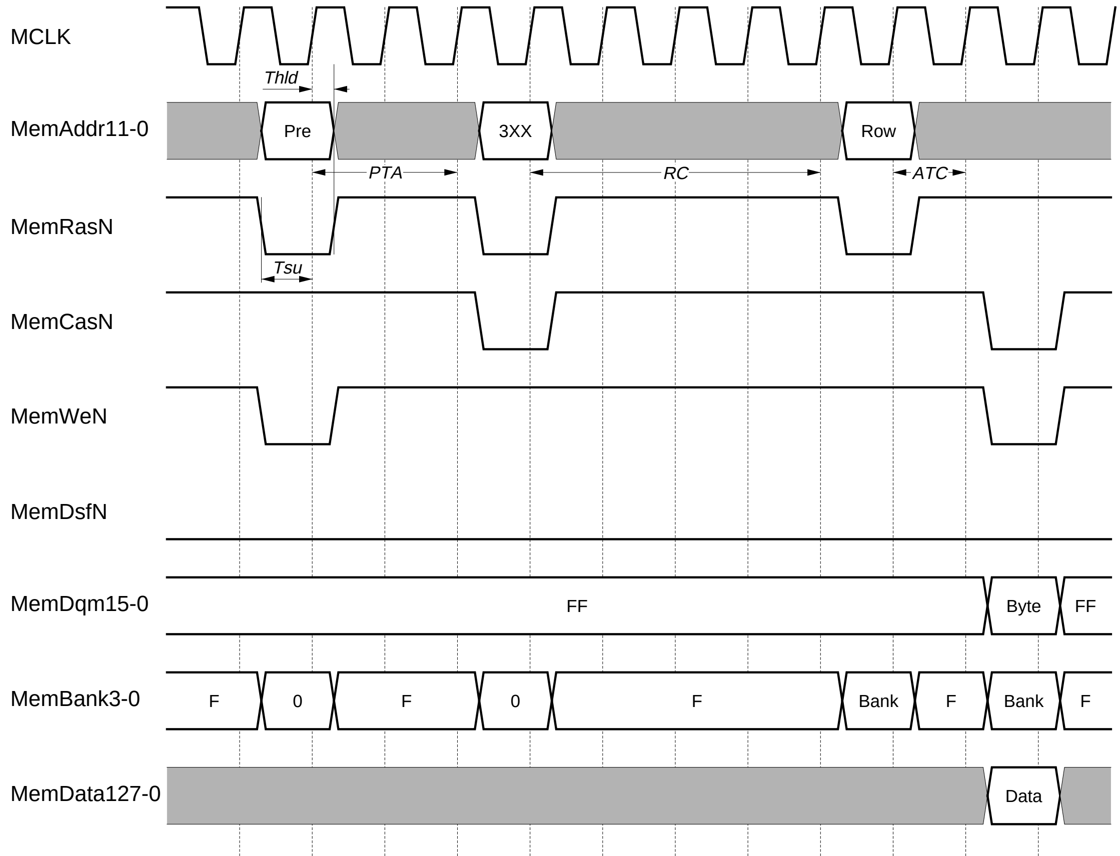
Single Write with Precharge @ PTA = 2, ATC = 1



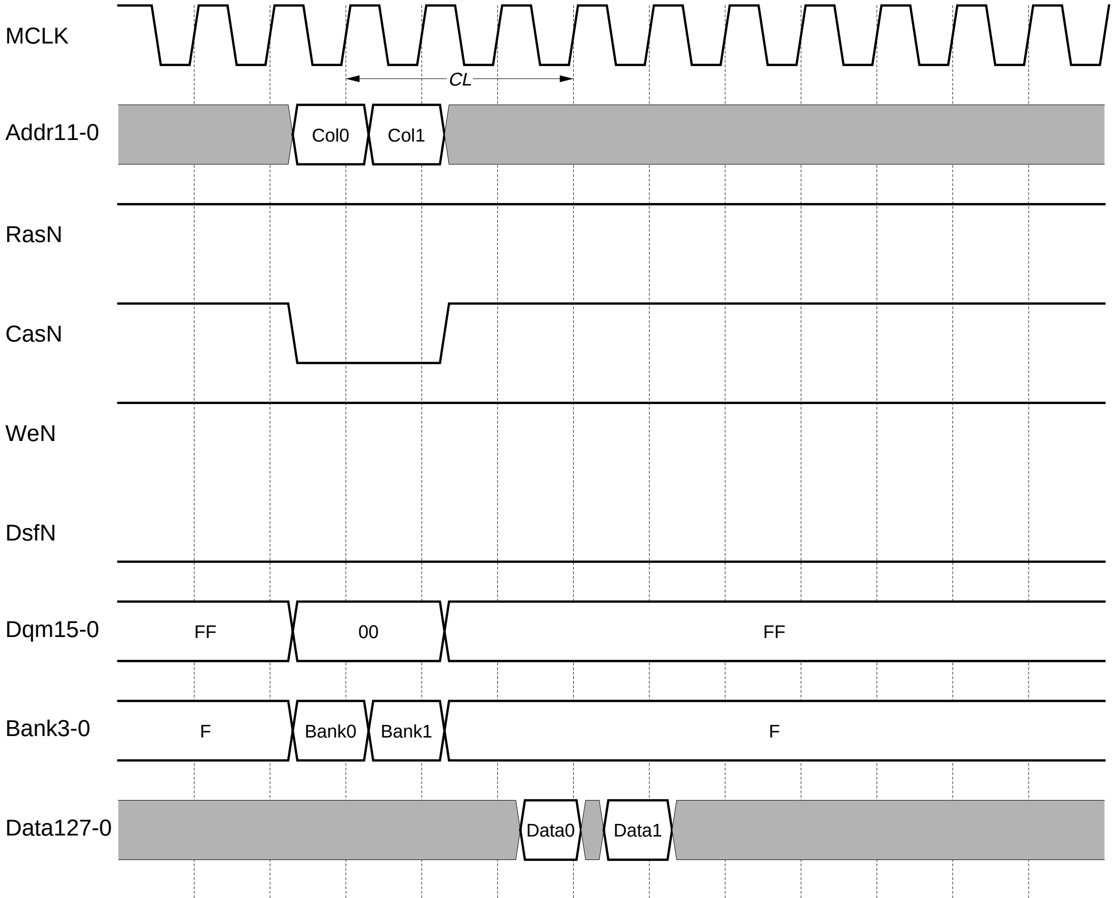
Multiple Writes to Same Bank @ PTA = 2, ATC = 1



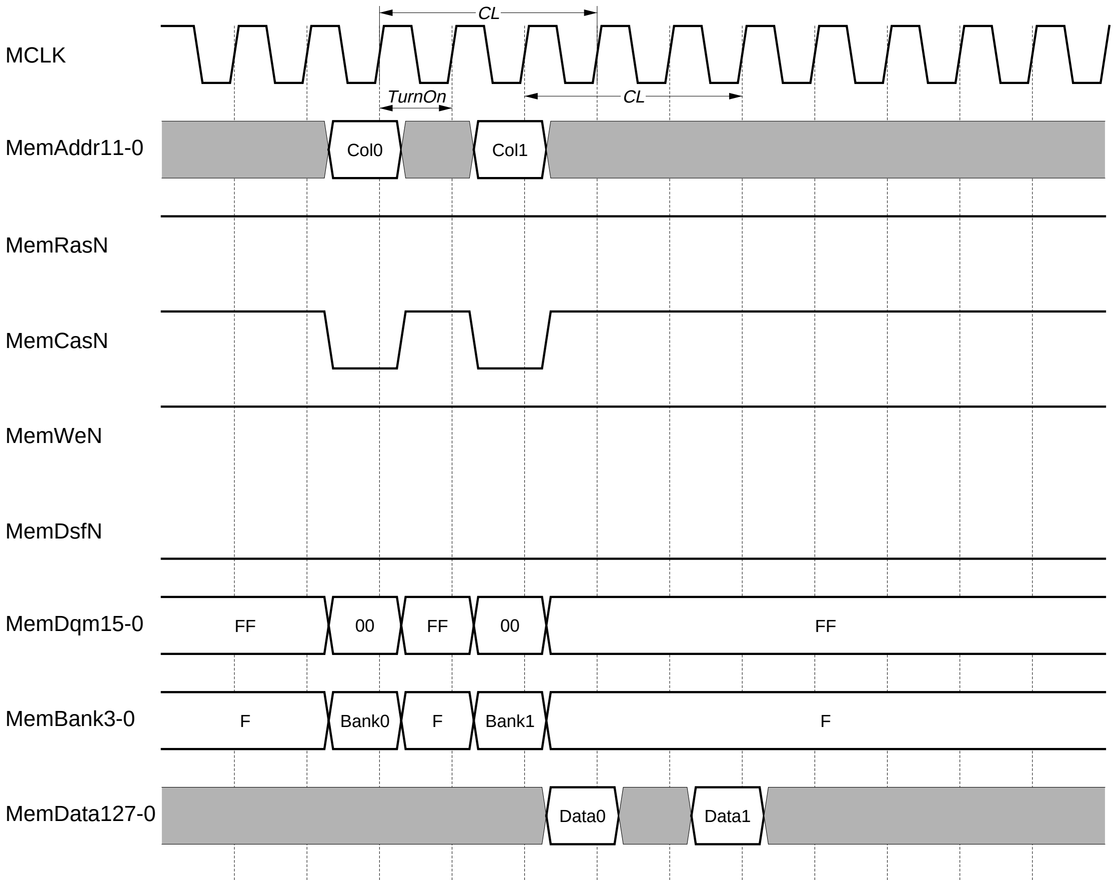
Refresh Followed by Access @ RC = 4, PTA = 2, ATC = 1



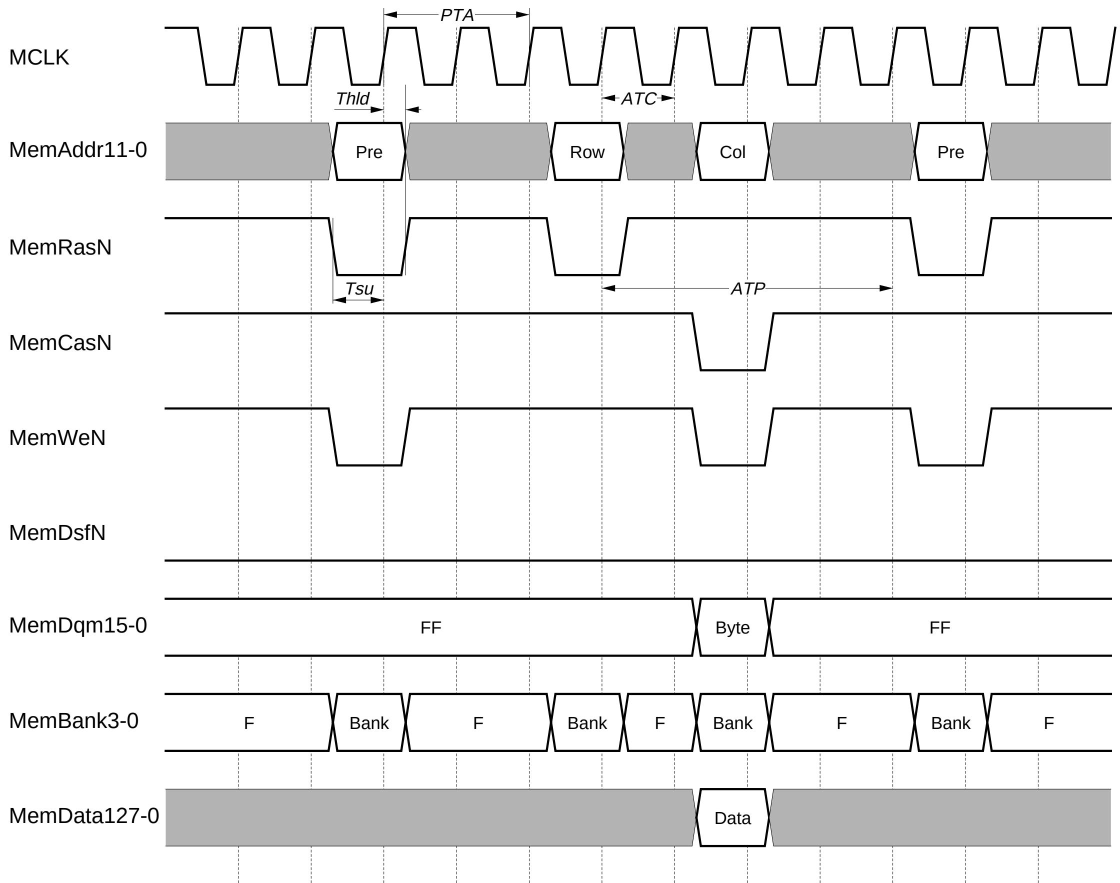
Multiple Reads From Different Banks @ TurnOn = 0, CL = 3



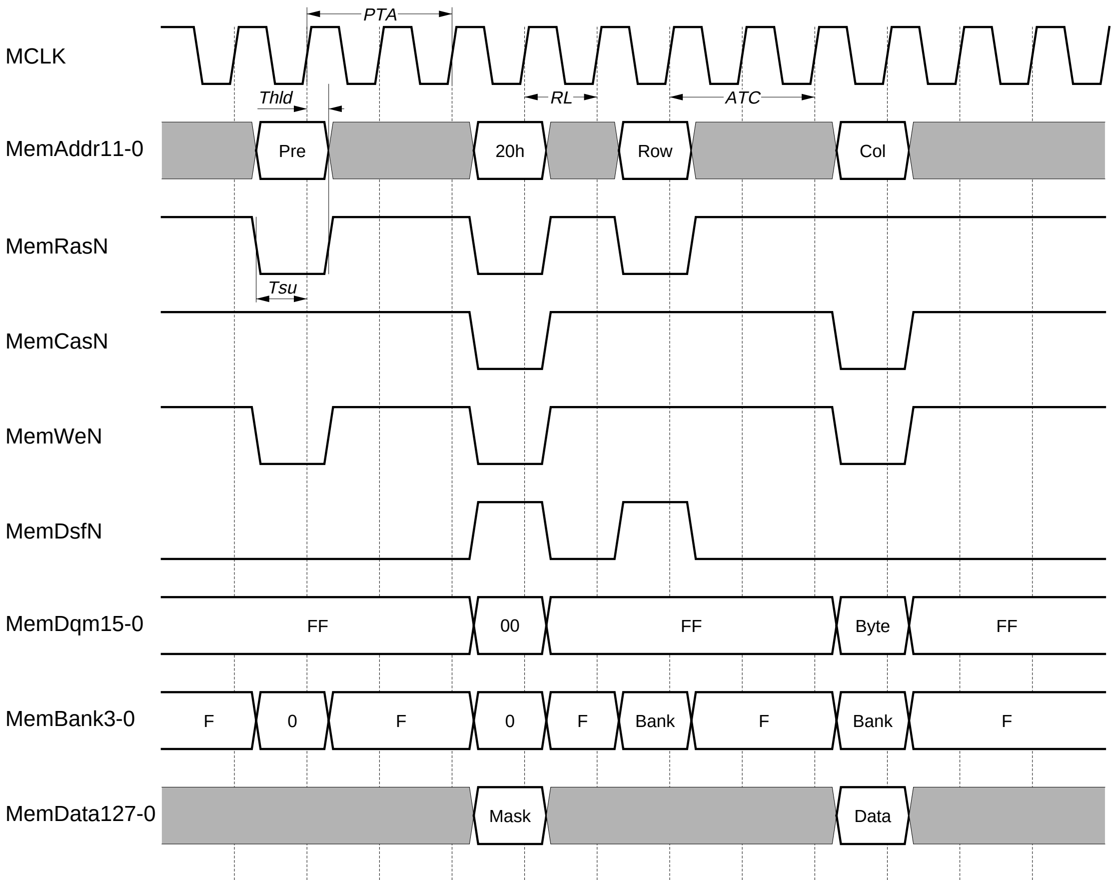
Multiple Reads From Different Banks @ TurnOn = 1, CL = 3



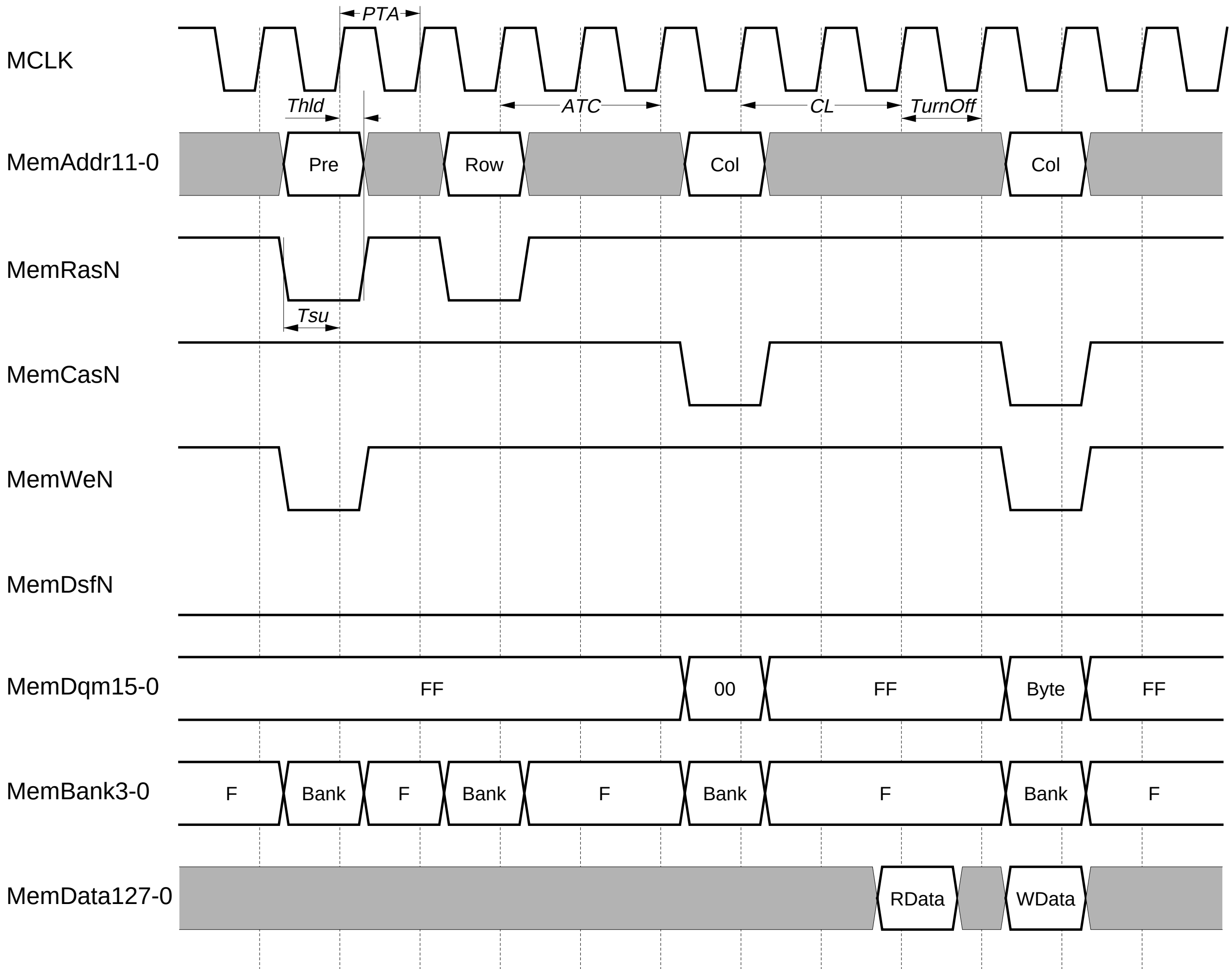
RAS Minimum Access Timing @ ATP = 4



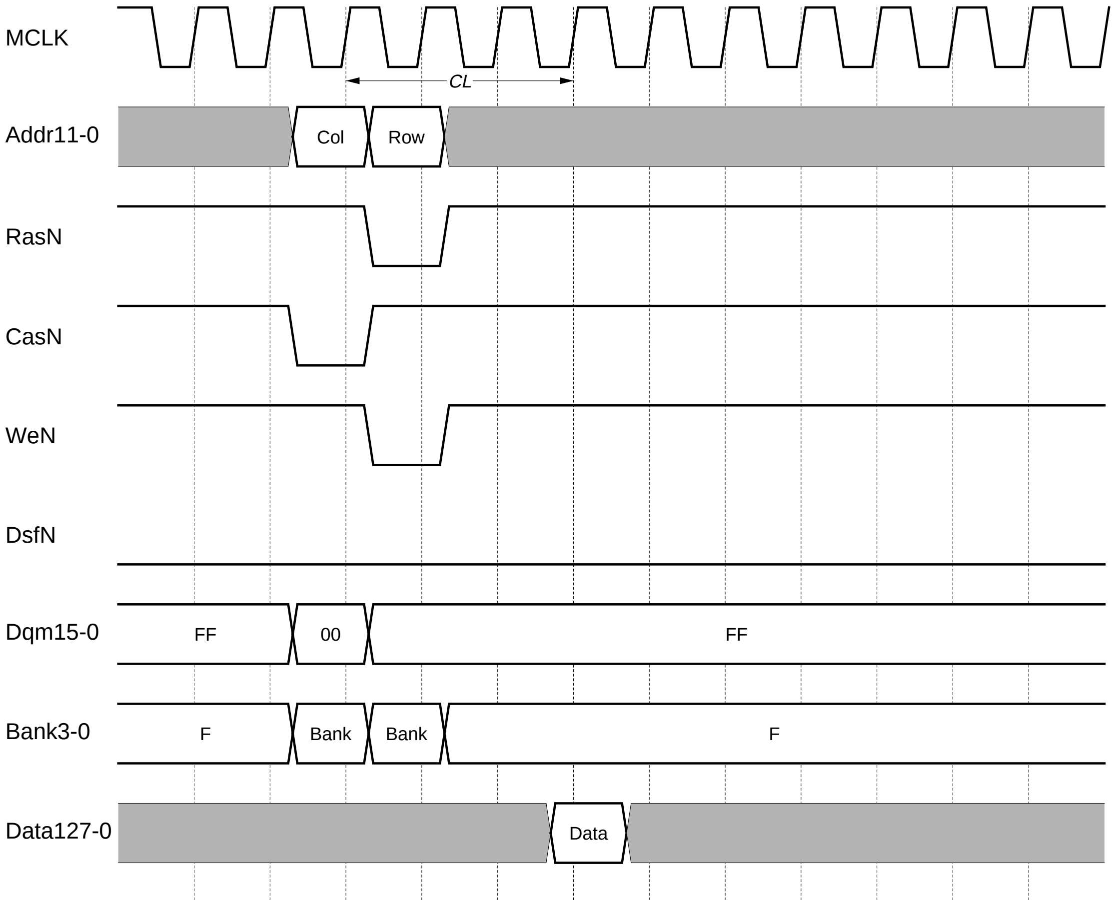
Mask Load Followed by Masked Write @ RL = 1



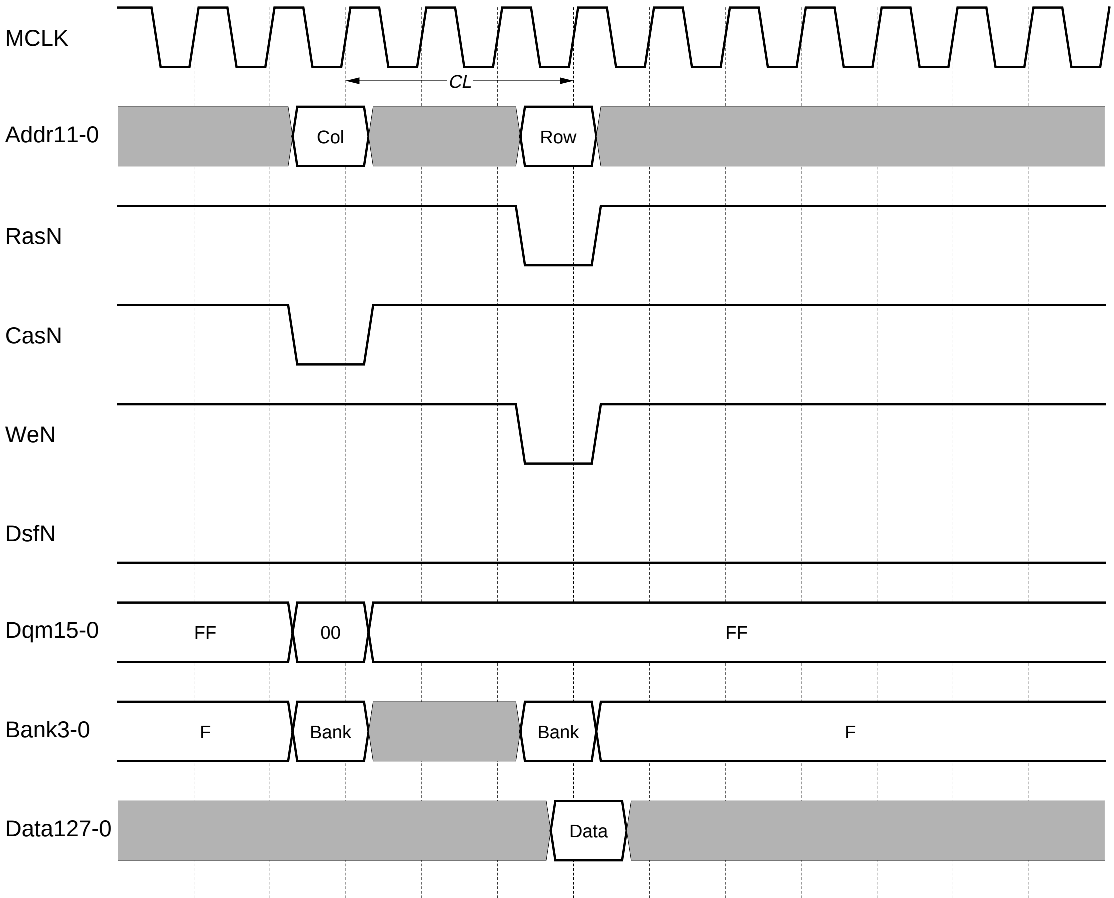
Read Followed by Write @ TurnOff = 1, CL = 2



Read Followed by PreCharge @ NoPrechargeOpt = 0 , CL = 3



Read Followed by PreCharge @ NoPrechargeOpt = 1, CL = 3



10

Reset Controls

At reset, certain signal pins are read and the values present, due to pull-ups or pull-downs, are used to initialized bits of particular registers.

Table 10-1 Reset Signal Pins

Name	pin	Description
BaseClassZero	VSAData(0)	1 = force PCI Bass Class Code to be zero
VGAEnable	VSAData(1)	1 = internal VGA subsystem present
VGAFixed	VSAData(2)	1 = enable VGA fixed address decoding
RetryDisable	VSAData(3)	1 = disable PCI Retry using "Disconnect-Without-Data"
ShortReset	VSAData(4)	1 = generate short "AReset" pulse (BusReset + 64 clocks)
AGP1XCapable	VSAData(5)	1 = AGP 1XCapable
SBACapable	VSBData(0)	1 = AGP Sideband Addressing Capable
SubsystemFromRom	VSBData(1)	1 = Load subsystem Vendor ID and SubsystemID from ROM . 0 = leave as reset values
AGP2XCapable	VSAData(6)	1 = AGP 2X Capable
AGP4XCapable	VSAData(7)	1 = AGP 4XCapable - this should never be set on a P3, unless 4X drivers are added
IndirectIOEnable	VSBData(2)	1 = Indirect IO accesses using BaseAddress 3 are enabled
WCEnable	VSBData(3)	0 = Upper half of region Zero is byte-swapped 1= Upper half of region Zero flagged internally as write-combined
PrefetchEnable	VSBData(4)	1 = Base Address registers 1 and 2 marked as prefetchable

A hard configuration pin also exists (Table 10-2).

Table 10-2 Hard Configuration Pin

Name	Pin	Description
PCIClk66	PCIClkSel	0 = up to 33MHz 1 = 66 MHz

Note: Note: During Power-up resets the external frequency reference must be powered at the same time as the Permedia3 or before it, to ensure normal operation.

11

Thermal Characteristics

The maximum junction temperature must be kept below T_j(max)and this can only be guaranteed by proper analysis of the operating environment and the thermal path between the die and the air surrounding it.

11.1 Device Characteristics

These are fixed characteristics of the device and are independent of the operating environment or the characteristics of any heatsink:-

Maximum Junction Temperature	$T_{j(max)}$	=	125 °C.
Maximum Power Dissipation	Pd(max)	=	6.7 Watts
Nominal memory clock frequency	f_{MClk}	=	125 MHz
Nominal core clock frequency	f_{KClk}	=	125 MHz
Junction to case resistance	θ_{jt}	=	5.5 °C/Watt

(Eq: 11-1)

11.2 Thermal Model

The formula used to calculate the junction temperature (T_j) is

$$T_j = T_a + Pd(\theta_{jt} + \theta_{cs} + \theta_{sa})$$
$$= T_a + Pd\theta_{ja} \qquad \text{(Eq: 11-2)}$$

Where:

T_j	=	Junction temperature (°C)
T_a	=	Ambient temperature (°C)
Pd	=	Power dissipation (Watts)
θ_{jt}	=	Junction to top of case thermal resistance (°C/Watt)
θ_{cs}	=	Case to Heatsink thermal resistance (°C/Watt)
θ_{sa}	=	Heatsink to Air thermal resistance (°C/Watt)
θ_{ja}	=	Total Junction to Air thermal resistance (°C/Watt)

(Eq: 11-3)

11.3 Cooling

PERMEDIA 3 should be operated with an attached heatsink or fan.

11.4 Operation with Heatsink

With a heatsink attached to the device the junction temperature will depend on θ_{cs} and θ_{sa} where θ_{cs} is the thermal resistance of the join between the heatsink and the case and θ_{sa} is the thermal resistance of the heatsink, which will be a function of system airflow. An ambient temperature of 40° C is assumed.

Heatsink to air thermal resistance			θsa
Maximum Junction Temperature	$T_{j(max)}$	=	125°C
Ambient Temperature	Ta	=	40°C
Maximum Power Dissipation	Pd(max)	=	6.7 Watts
Junction to case resistance	θjt	=	5.5°C/Watt
Heatsink to case resistance	θcs	=	1.0°C/Watt
(EG 7655 epoxy - see below)			

then:

$$\theta sa \le [(125 - 40)/6.7] - 5.5 - 1.0$$

$$\le 6.2°C/Watt. \quad \text{Eq: 11-4}$$

In this example a heatsink must be chosen which has a thermal resistance figure of no greater than 6.2°C/Watt at an airflow matching the expected airflow in the system..

11.5 1.1 Heatsink Attachment

The following method has been approved for the purpose of attaching a heatsink directly onto the copper surface of the SBGA package:

Thermally conductive epoxy using either Loctite Output 315 with Loctite 7386 or type EG 7655 from A.I. Technology Inc. The thickness of the epoxy layer should be between 0.05mm and 0.15mm with 95% coverage of the contact area.

Typical achievable ?cs using this method is 1.0 ° C/Wat

12
Electrical Characteristics

12.1 Absolute Maximum Ratings

Junction Temperature	125°C
Storage Temperature	-65°C to 150°C
VCC2.5 DC Supply Voltage	2.8V
VCC3.3 DC Supply Voltage	3.8V
I/O Pin Voltage with respect to GND	-0.5V to VCC3.3 + 0.5V

12.2 DC Specifications

Symbol	Parameter	Min	Max	Unit
VCC3.3	Supply Voltage	3.15	3.45	V
VCC2.5	Supply Voltage	2.25	2.75	V
LPIN	Pin Inductance		10 (sig); 8 (pwr)	nH
ICC (3V)	Power Supply Current		1	A
ICC (2.5V)	Power Supply Current		1.4	A

12.2.1 PCI Signal DC Specifications

Symbol	Parameter	Min	Max	Unit
V_{PIL}	Input Low Voltage		0.8	V
V^{PIH}	Input High Voltage	2.0		V
V_{POL}	Output Low Voltage		0.5	V
V_{POH}	Output High Voltage	2.4		V
I_{PIL}	Input Low Current		-20	uA
I_{PIH}	Input High Current		+20	uA
C_{PIN}	Input Capacitance		10	pF
C_{CLK}	PCI Clock Input Capacitance		10	pF
C_{IDSEL}	PCI Idsel Input Capacitance		8	pF

12.2.2 Non-PCI Signal DC Specifications

Symbol	Parameter	Min	Max	Unit
V_{IL}	Input Low Voltage		0.8	V
V_{IH}	Input High Voltage	2.0		V
V_{OL}	Output Low Voltage		0.5	V
V_{OH}	Output High Voltage	2.4		V
I_{IL}	Input Low Current		+10	uA
I_{IH}	Input High Current		-10	uA
I_{IHPD}	Pulldown Input High Current		250	uA
I_{ILPU}	Pullup Input Low Current		250	uA
C_{IN}	Input Capacitance		10	pF

12.3 AC Specifications

Pin Name	Capacitive Load
MADD[9:0]., MCAS[1:0], MDSF[1:0], MRAS[1:0], MWE[1:0].	80pF
PCIAD[31:0], PCICBEN[3:0], PCIPar, PCIFrameN,PCIIRdyN, PCITRdyN, PCIStopN, PCIIdsel, PCIDevselN, PCIReqN, PCIGntN, PCIIntAN ,AGPPipeN, AGPRbfN, AGPSBA[7:0],	50pF in PCI 33 system 10pF in AGP system
MBANK[3:0], MBYTE[7:0], MEMCKE, MEMCKOUT[1:0], VidDDCClk, VidDDCData, VidRightEye, VidHSYNC, VidVSYNC,VSAResetN,VSBResetN,RenderSyncN	50pF
MDAT[63:0].	40pF
ROMSelectN, ROMWEN, SBClk, SBData, VSAData[7:0], VSBData[7:0], VSCtl[7:0], VSGPChipSelectN, VSGPDataAckN, VSGPDataStrobeN, VSGPReadWriteN.	30pF

12.3.1 Clock Timing

Symbol	Parameter	Min	Max	Units	Notes
T_{PCyc}	PCIClk Cycle Time	15	-	ns	
T_{PHigh}	PCIClk High Time	-	-	ns	
T_{SLow}	PCIClk Low Time	-	-	ns	
T_{MCyc}	MClkin Cycle Time	8	-	ns	
T_{MHigh}	MClkin High Time	-	-	ns	
T_{MLow}	MClkin Low Time	-	-	ns	
T_{SCyc}	SClkin Cycle Time	15	-	ns	
T_{SHigh}	SClkin High Time	6	-	ns	
T_{SLow}	SClkin Low Time	6	-	ns	
T_{DCyc}	DClk Cycle Time	4	-	ns	
T_{DHigh}	DClk High Time	-	-	ns	
T_{DLow}	DClk Low Time	-	-	ns	

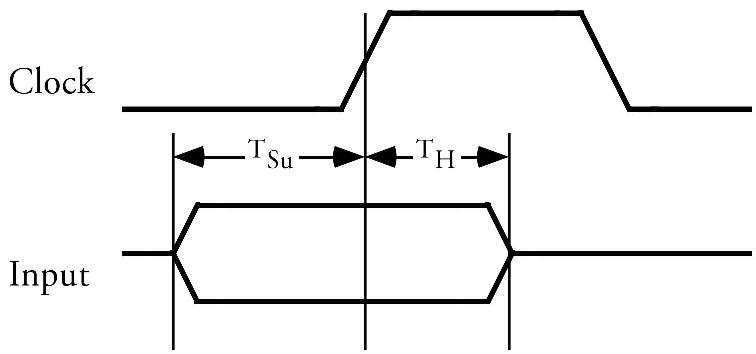


Figure 12.1 Input Timing Parameters

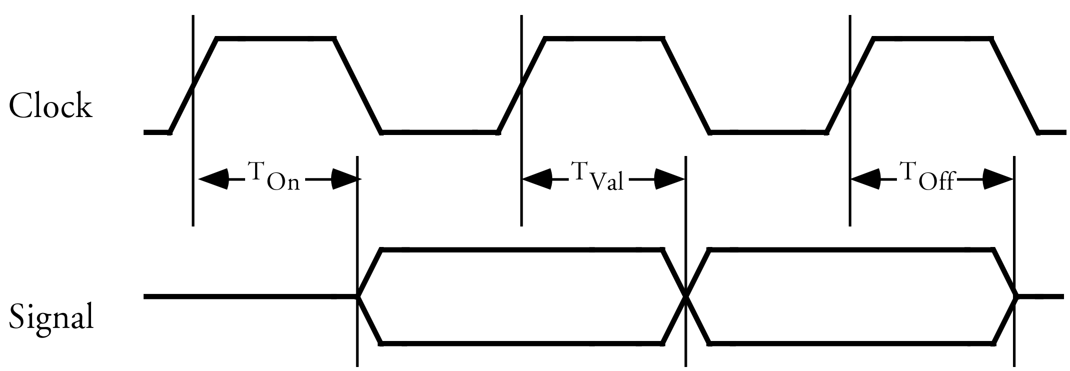


Figure 12.2 Output Timing Parameters

12.3.2 PCI Clock Referenced Input Timing

Parameter	T_{Su} Min	T_H Min	Units	Notes
PCIAD(31:0), PCICBEN(3:0), PCIPar, PCIFrameN, PCIIRdyN, PCITRdyN, PCIStopN, PCIIdsel, PCIDevselN, AGPSt0-2	5	0	ns	
PCIGntN	5	0	ns	
PCIRstN	7	0	ns	1

Note 1: PCIRstN is resynchronised internally. The timings given, when met, ensure that the reset is detected in the current cycle.

12.3.3 PCI -Referenced Output Timing

	T_{Val}		T_{On}		T_{Off}			
Parameter	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Units**	**Notes**
PCIAD(31:0), PCICBEN(3:0), PCIPar, PCIFrameN, PCIIRdyN, PCITRdyN, PCIStopN, PCIIdsel, PCIDevselN	2	11	2	11	2	11	ns	
PCIReqN	2	12					ns	
PCIIntAN	2	11					ns	1

Note 1: Timings given are for falling edges of the open drain signal. Rise times are dependent on the external pull-up resistor.

12.3.4 AGP Referenced Output Timing

	TVal		TOn		TOff			
Parameter	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Units**	**Notes**
PCIAD(31:0), PCICBEN(3:0), PCIPar, PCIFrameN, PCIIRdyN, PCITRdyN, PCIStopN, PCIIdsel, PCIDevselN	1.5	6	1.5	6	1.0	14	ns	
PCIReqN	1.5	6					ns	
PCIIntAN	1.5	6					ns	1

Note 1: Timings given are for falling edges of the open drain signal. Rise times are dependent on the external pull-up resistor.

12.3.5 MEMCKOUT Referenced Input Timing

All timings below are with respect to MEMCKOUT, which is a delayed version of MClk.

Parameter	TSu Min	TH Min	Units	Notes
MDAT[63:0]	1	3	ns	

12.3.6 MEMCKOUT Referenced Output Timing

All timings below are with respect to MEMCKOUT, which is a delayed version of MClk.

	TVal		TOn		TOff			
Parameter	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Units**	**Notes**
All memory control, data and address lines		8.5					ns	

13

Index

A

B

C

D

E

F

G

H

R

S

T

U

V