GLINT R4®

Reference Guide Volume III -

Graphics Registers

PROPRIETARY AND CONFIDENTIAL INFORMATION

3D*labs*®

GLINT R4®

Reference Guide Volume III -

Graphics Registers

PROPRIETARY AND CONFIDENTIAL INFORMATION

Issue 2

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.

© Copyright **3D***labs* Inc. Ltd. 1999. All rights reserved worldwide.

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
160.5.3	1	06 October 99	Creation
160.5.3	2	05 December 99	ReadMonitorMode, SetDeltaPort; consistent use of GLINT R4 nomenclature; adding app 55 new R4-only registers plus updates, delete P4-only registers; update TextureCoordMode note, add T1Start, add TextureData,add to TextGlyph, add FBHardwareWriteMask; delete FeedbackX, FeedBackY; ReadMonitor logic ops; Reinstate Download Address, Download Data; edit vertex registers to include deprecated fixed format for R4 delta; DMARectangleRead, DMAContinue not readbackable, DeltaMode TargetChip value fixed, Color reg reinstated, Context Dump/restore updated, removed priv. regs FBData & FBSourceData, R4 changes to FilterMode; LBClearDataU and ScanLineOwnership reinstated, other register updates; removed remaining instances of MultiRXBlit bits in Config2D, FillConfig2D; Removed dKsDx & other speculars; deleted FeedbackX and Y from Xref chapter, deleted from Xref Drawpoint, Provoking vertex, TextureLOD, Drawline & Linecoord; added Begin, End to Xref;

Table of Contents

5
Graphics Registers

This chapter lists PERMEDIA4 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.

AALineWidth

Name	Type	Offset	Format
AALineWidth	R4 Delta ***Control register***	0x94C0	Float

Bits	Name	Read	Write	Reset	Description
0..31	LineWidth	✓	✓	x	Floating point 32 bit integer

Notes: Defines the width of antialiased lines stored as a floating point number using the absolute value of the width. Lines with a zero width are drawn XXX and lines with a negative width are drawn as a line of the same positive width . Antialiased lines are drawn as rectangles aligned to the direction of the line.

AAPointSize

Name	Type	Offset	Format
AAPointSize	R4 Delta ***Control register***	0x94A0	Float

Bits	Name	Read	Write	Reset	Description
0…31	Point size	✓	✓	x	Point size value

Notes: Defines the width of antialiased points. In theory any size antialiased point can be defined but in fact the Point Table restricts the diameter of antialiased points from 0.5 to 16.0 in steps of 0.25 (when the antialiasing quality is 4x4) or 0.25 to 8.0 in steps of 0.125 (for 8x8 quality). Points with a zero size draw a single fragment and points with a negative size draw a point of the same positive size. N.B. Alpha blending modes must be set up first.

The point size table is enabled in the *UsePointTable* field of the *Render* register. The table is 32 entries deep by 4 bits wide. The unsigned delta values in the table are held as one bit integer and 3 bits fraction. From the host's view the table is organised as four 32 bit words so the overhead of downloading when the point size changes is minimal. Only the parts of the table needed for a particular point size need to be loaded. The format of the points table is as follows:

Point Size Table

		24		16		8		0
PointTable0	P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0
PointTable1	P_{15}	P_{14}	P_{13}	P_{12}	P_{11}	P_{10}	P_9	P_8
PointTable2	P_{23}	P_{22}	P_{21}	P_{20}	P_{19}	P_{18}	P_{17}	P_{16}
PointTable3	P_{31}	P_{30}	P_{29}	P_{28}	P_{27}	P_{26}	P_{25}	P_{24}

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.
5…7	DestBlend	✓	✓	x	This field defines the destination blend function to use.
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
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

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

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.
For information on the implementation of specific belnd interpolations etc. refer to the *Permedia4 Programer's Guide*, volume II, Source Blending Functions

The logic operator versions of the mode command behave the same way as the command 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 effected by this swap and the result is always on internal R channel.

The format to use is held in the *AlphaBlendColorMode* register. Note that in OpenGL the 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

				Internal Color Channels			
	Format	**Color Order**	**Name**	**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

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

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
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.

For information on the implementation of specific belnd interpolations etc. refer to the *Permedia4 Programer's Guide*, volume II, Source Blending Functions

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.

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 R4'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
BasePageOfWorkingSetHost	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).

Begin

Name	Type	Offset	Format
Begin	R4 Delta	0x9590	Bitfield
	Command Rregister		

Bits	Name	Read	Write	Reset	Description
0	AreaStipple Enable	✗	✔	x	When compatible with primitive type, 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	When compatible with primitive type, 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	Overridden when incompatible with the primitive type. Generated by the vertex unit or supplied by the Gamma **GeomTriangle** or **GeomQuad** registers.
3	FastFillEnable	✗	✔	x	Ignorced, disabled.
4, 5	Unused	0	0	x	

6, 7	Primitive Type	✗	✔	x	Generally overridden (except where the *Type* and PolyMode settings are compatible) since support for *Gamma* types in the *Type* field is much more extensive
8	Antialiase Enable	✗	✔	x	Qualifies the Antialiase enable held for each primitive by the Delta unit. When both are 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.
9	Antialiasing Quality	✗	✔	x	Ignored - the information is held in the Delta unit on a per-primitive basis.
10	UsePoint Table	✗	✔	x	Ignored - generated locally when antialiasing points
11	SyncOnBit Mask	✗	✔	x	Ignored - disabled
12	SyncOnHost Data	✗	✔		Ignored - disabled
13	TextureEnable	✗	✔	x	Passed through Delta unit, also enables texturing in Geometry and Lighting units.
14	FogEnable	✗	✔	x	Passed through Delta unit, also enables fog calculation to be performed in the Geometry unit.
15	Coverage Enable	✗	✔	x	Ignored on input and Set on output if the primitive is to be antialiased.
16	SubPixel Correction Enable	✗	✔	x	Passed through Delta unit but modifies rasterizer setup. When set enables the sub pixel correction of the colour, 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	Passed through R4 Delta with no effect. This bit, when clear, indicates the writes are to use the constant colour found in the previous FBBlockColour message. 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	dErr	✗	✔	x	This field holds the subpixel error to apply on the first scanline. The subpixel correction occurs when the SubPixelCorrection message is sent by the rasteriser and the value here just allows the multiplication to be pipelined.
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	Primitive type	✗	✔	x	This field sets up the primitive type to process on receiving each new vertex. It has the following values: 0=null 6=TriangleStrip 1=points 7=TriangleFan 2=lines 8=Quads 3=lineLoop 9=QuadStrip 4=lineStrip 10=Polygon 5=Triangles

Notes: The **Begin** command, as in OGL, starts a series of vertices by specifying the primitive type and various enables. The primitive type is held in the upper 4 bits and the lower bits hold the same fields as the **Render** command. The R4 **Begin** command is integrated with the Gamma Delta **Begin** command when the R4 is acting as rasterizer for a Gamma geometry accellerator.

BitMaskPattern

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

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* bit 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 *Control register*	0x87B0	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 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	Color *Control registers*	0x8750	Bitfield

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

Notes: The Color register is supported primarily for backwards compatibility. It may be used to provide input to the Rasterizer during image download but cannot be written to as an alternative to **ConstantColor** for rendering flat shaded depth-buffered primitives.

Note: The Color register cannot be saved dring context switch.
The Color DDA must be disabled when using the Color register.

For CI mode coloring the CI is placed in bits 0...7. If less than 8 bits it should be shifted to the MSB end and the LSBs set to 0.

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

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[8]	Write	Reset	Description
0	Enable	✓	✓	x	This bit, when set, causes the current color to be generated.
1	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.
2…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	0	0	x	

Notes:

[8] 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	OpaqueSpan	✓	✓	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	LUTModeEnable	✓	✓	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[9]. 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.

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	20
1	Geometry	✕	✔	x	Delta unit state	68
2	Matrices	✕	✔	x	unused	0
3	Material	✕	✔	x	unused	0
4	Lights0_7	✕	✔	x	unused	0
5	Lights8_15	✕	✔	x	unused	0
6	RasterPos	✕	✔	x	unused	0
7	CurrentState	✕	✔	x	unused	0
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	0
11	RasterizerState	✕	✔	x	General setup of the rasterization units	225
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

[9] 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.

16	TextureManagement	✗	✔	x	State used for logical texturing (virtual texturing)	9
17…31	Reserved	0	0	x		0

Notes: This command forces the R4 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	20
1	Geometry	✗	✔	x	Delta unit state	68
2	Matrices	✗	✔	x	unused	0
3	Material	✗	✔	x	unused	0
4	Lights0_7	✗	✔	x	unused	0
5	Lights8_15	✗	✔	x	unused	0
6	RasterPos	✗	✔	x	unused	0
7	CurrentState	✗	✔	x	unused	0
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	0
11	RasterizerState	✗	✔	x	General setup of the rasterization units	225
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		0

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.

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	The bitfield is retained for backwards compatibility but in R4 the value must be initialized to 1.
2, 3	DepthFormat	✓	✓	x	This field defines the depth format and hence the final format of the depth parameters to be written into the R4. 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 commands.
6	Smooth Shading Enable	✓	✓	x	When set enables the color calculations. Only effective when the Draw* commands are used.
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* bit 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 R4. This only affects 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	Reserved
17	Reserved	0	0	x	Reserved

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

18	Reserved	✓	✓	x	Reserved
19	Bias Coordinates	✓	✓	x	When set, adds the Xbias and Ybias values to the x and y coordinates. Allows (e.g.) remval of an OGL viewport bias or conversion of a windows-relative to screen-relative co-ordinate system. 0 = off, 1 = on
20	Reserved	✓	✓	x	Reserved
21	Reserved	✓	✓	x	Reserved
22	FlatShading Method	✓	✓	x	Specifies which shading method the Delta unit should direct the ColorDDA to employ: 0 = Use ConstantColor register value 1 = Use DDA with zero gradients
23...31	Reserved	0	0	x	Reserved

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 Control Register	0x8818	Bitfield
DitherModeAnd	Global	0xACD0	Bitfield Logic Mask
DitherModeOr	Global	0xACD8	Bitfield Logic Mask

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	

Notes: Dithering controls color formatting. The dither function converts the internal color format into the framebuffer color information format.

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

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 r*egister 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.

Note: Feedback DMA must run as external DMA only.

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 R4 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 R4 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 R4 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 R4. 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 R4 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. R4 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; DMARectangle ReadTarget**.

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 R4. 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 R4. 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 R4. 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	1 = data is right justified and any unused bits (in the most significant end of the word) are set to zero. 0 = data read from the Host Out FIFO is packed. N.B. this is the inverse of the bit setting in **DMARectangleRead**
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 R4 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 R4 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: 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. P4 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: Erratum PEREN009; **DMARectangleReadAddress***;* **DMARectangleReadLinePitch***;* **DMARectangleReadTarget**

DMARectangleWriteAddress

Name	**Type**	**Offset**	**Format**
DMARectangleWrite Address	Input *Control register*	0xA9D0	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 write during a rectangular DMA transfer from R4 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 R4 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: 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.

DrawLine01

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:
- *DrawLine0* sets up and renders a line from vertex 0 to vertex1, *DrawLine1* draws a line from vertex 1 to vertex 0. The vertices are loaded separatel

DrawLine10

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.
- DrawLine01 draws a line from vertex 0 to vertex1, DrawLine10 draws a line from vertex 1 to vertex 0.

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	1 = Enable 0 = Disable Enables texturing of the fragments produced during rasterisation. Used primarily to disable texture for specific primitives. C.f. **DeltaMode** register.

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	RejectNegative Face	✗	✔	x	Reject faces with negative area if backface cull is enabled
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. *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. Register known as **TexelCoordUV** in MS, Permedia2 and earlier chipsets.

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. Register known as **TexelCoordU** in MS, Permedia2 and earlier chipsets.

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.

End

Name	Type	Offset	Format
End	Delta *Control register*	0x9598	Fixed point

Bits	Name	Read	Write	Reset	Description
0…31	Fraction	✓	✗	x	

Notes: Terminates a Begin/End pair defining a primitive to render. **End** simply consumes the message since it is not used subsequently. See **Begin**.

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 R4 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 register holds 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.

Note: When R4 is used in conjunction with G1/G2, this tag is reserved for use by the Gamma and does not affect the rasterizer(s).

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 foreground pixels. Readback returns the data in *FBBlockcolor0*.

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 the Color unit. 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).

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	E0 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	GLINT 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 times 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 R4.
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 R4.
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…15	Layout1	✔	✔	x	Selects the layout of the pixel data in memory for buffer 1.

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

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 registerr***	0x8AC0	

Bits	Name	Read	Write	Reset	Description
0…31	Write mask	✓	✓	x	32 bit mask

Notes: This message 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 a framebuffer hardware writemask is configured. 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 registerr*	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.

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
FBSourceReadBufferOffset	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: Holds the offset added to the fragment's coordinate for the source buffer. The new coordinates are used for address calculations. The 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 R4.
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 R4.
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 *Control register*	0xB040, 0xB048, 0xB050, 0xB058	Integer

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 FBDestRead are enabled then writes 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 FBDestRead Mode.
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 R4.

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

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 R4.
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.

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* - reserved on R4
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	ActiveTag	✔	✔	x	When set allows the **PrepareToRender, SuspendReads, ActiveStepX** and **ActiveStepYDomEdge** tags to be written into the output FIFO.
1	ActiveData	✔	✔	x	When set allows the **PrepareToRender, SuspendReads, ActiveStepX** and ActiveStepYDomEdge data to be written into the output FIFO.
2	PassiveTag	✔	✔	x	When set allows the PassiveStepX and PassiveStepYDomEdge tag to be written into the output FIFO.
3	PassiveData	✔	✔	x	When set allows the PassiveStepX and PassiveStepYDomEdge data to be written into the output FIFO.
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.

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

15	Remainder Data	✔	✔	x	When set allows any data not covered by the categories in this table to be written into the output FIFO.
16…17	ByteSwap	✔	✔	x	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
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	ExternalDMA Controller	✔	✔	x	This bit, when set, changes the protocols which are used when interacting with an external DMA controller (e.g. Gamma II).. In this case the protocols are limited to what a PCI bus can support (i.e. limited to 32 bits). When clear an internal DMA controller is used (i.e. in P3) so a more efficient protocol can be used. This bit only has an effect when run length encoding is used.
22...31	Reserved	0	0	x	Reserved

Notes: This register can only be updated if the *Security* register is set to 0.

Run Length Encoded Data

Image data frequently contains runs of the same pixel data so lends itself to run length encoding so less data is needed to describe the image. This does not speed up the image upload from the core's viewpoint as it still needs to read the data, but it reduces the amount of data transferred over the PCI bus, and potentially the host effort in processing/copying the image data.

When run length encoding is enabled then *any* data (but not tags) which would have been written into the FIFO is accumulated into the run length while it matches the 32 bit run length value. The accumulated run length is written to the FIFO when:

- The new 32 bit word is different from the run being encoded.
- A new scanline is started.
- The end of the primitive occurs.

The amount of data produced during the run length encoding is not known when the DMA controller is set up so an alternative mechanism is used to tell the DMA controller the upload data has finished. This is done by using the EndOfFeedback command and when this is detected in the DMA controller the DMA can be terminated as all the data will have been received.
The run length data written into the FIFO depends on where the DMA controller resides. There are two options:

- The DMA controller is external (i.e. in Gamma). In this case the tag and data are needed so the DMA controller can track what is going on and know when to finish. In this configuration the tag and data stream shown in the table below is written to the Host out FIFO:

- The DMA controller is internal (i.e. on the same die as in the case of P3). In this case a

more efficient protocol can be adopted so the tags are not included in the FIFO and an extra bit (on the FIFO width) is used to tell the DMA controller to finish. This bit is only set when an EndOfFeedback tag is received *during RLE processing*. This allows run length data to be uploaded at twice the rate available with an external DMA controller.

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.

Table 5.2 External DMA Controller FilterMode data

Gamma Name	Tag Value	Data
Feedback_X	0x1F1	run length (max = 0xffffff)
Feedback_Y	0x1F2	run length value
EndOfFeedback	0x1FF	0 (not used)

There are four entries in the output FIFO per run length in the order given in the above table. A zero run length will never be entered into the FIFO. The **EndOfFeedback** tag and null data identify the end of the upload.

In normal operation the message filtering is set up so no tags are written to the FIFO and only the data pertinent for the upload is written. Note the tags are internally generated and are essential for any software or the output DMA controller to track and interpret the FIFO contents. If the output DMA controller is used in 'Feedback' mode then the tags are filtered out and just the data is written into memory.

Note: The Output DMA controller can potentially hang the system when it is in feedback mode and the RunLengthEncodeData *bit is not set. In this situation the tag will not be detected by the DMA controller, hence the software will not be informed the upload has finished. The graphics core continues to function but the Output DMA controller loops forever discarding data once the buffer count has expired.*

If the software is running with a time-out the Output DMA controller can be recovered by setting the *RunLengthEncodedData* bit and sending an **EndOfFeedback** message. Note that this situation should not legitimately occur as the only time when the Output DMA controller is used in feedback mode is when the amount of data to upload is not known and in an R4 only system this will only occur with run length encoded data. The security features of the Host In Unit will prevent accidental modification of the **FilterMode** register and initiation of Output DMA.

A GLINT R4 with Gamma does not have this problem as the internal DMA controller is not used.

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 use as a basis for interpolation..

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 fog operation takes the Color value from a fragment and interpolates between the Color value (from the primitive's step registers) and a fog Color (from the **FogColor** register) using an internally generated interpolation coefficient. The interpolation coefficient is calculated on a per fragment basis using a DDA unit, and optionally remapped via a look up table. If the fog is disabled (either from the mode register or from the *FogEnable* bit in the **Render** command) then the fragment's Color is passed on to the next unit 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.

[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	0xB0C0	Integer
	Control register		

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 interpolation coefficient is used to blend the fragment's color with the color in the **FogColor** register. 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

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

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.

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.

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 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	Localbuffer *Control register*	0xB558	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Address	✓	✓	x	32 bit integer value

Notes: This register holds the 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.

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	E0 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	GLINT 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.

LBDestReadMode
LBDestReadModeAnd
LBDestReadModeOr

Name	Type	Offset	Format
LBDestReadMode	Localbuffer	0xB500	Bitfield
LBDestReadModeAnd	Localbuffer	0xB580	Bitfield Logic Mask

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

LBDestReadModeOr	Localbuffer *Control registers*	0xB588	Bitfield Logic Mask

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 R4.
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 R4.
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.

LBReadFormat

Name	Type	Offset	Format
LBReadFormat	Localbuffer *Control register*	0x8888	Bitfield

Bits	Name	Read	Write	Reset	Description

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

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	Reserved	0	0	x	
15…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	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. Note: LB ReadFormat register definition has changed to allow more flexible sizing and positioning of the GID and stencil fields.

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 *Control registers*	0xB5A8	Bitfield Logic Mask

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 R4.
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 R4.

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

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.

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 R4.
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 R4.
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.

LineMode
LineModeAnd
LineModeOr

Name	Type	Offset	Format
LineMode	Delta	0x94A8	Bitfield
LineModeAnd	Delta	0xAAF0	Bitfield Logic Mask
LineModeOr	Delta	0xAAF8	Bitfield Logic Mask

Control registers

Bits	Name	Read [24]	Write	Reset	Description
0	StippleEnable	✔	✔	x	This field, when set, enables the stippling of lines. It only effects wide lines or antialiased line set up. This will normally be the same value as the Enable field in the LineStippleMode GLINT register.
1…9	RepeatFactor	✔	✔	x	This field holds the positive repeat factor for antialiased stippled lines. This will normally be the same value as the RepeatFactor field in the LineStippleMode GLINT register. The repeat factor stored here is one less than the desired repeat factor.
10…25	StippleMask	✔	✔	x	This field holds the stipple pattern to use for antialiased lines. This will normally be the same value as the StippleMask field in the LineStippleMode GLINT register.
26	Mirror	✔	✔	x	This field, when set, will mirror the StippleMask before it is used for antialiased lines. This will normally be the same value as the Mirror field in the LineStippleMode GLINT register.
27	AntialiasEnable	✔	✔	x	This field, when set, enables antialiasing of lines. This is qualified by the AntialiasEnable field in the Begin message.
28	Antialiasing Quality	✔	✔	x	This field defines the quality of antialiased points: 0 4x4 1 8x8
29	DrawLastPoint	✔	✔	x	This field defines if the last point on a non antialised line should be drawn. The normal state for OpenGL is not to draw, while in D3D this is an option.

Notes: Defines how wide and altialiased lines are processed - applies only to lines rendered using RenderLine. 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.

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

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.

LineWidth

Name	Type	Offset	Format
LineWidth	Delta	0x94B0	Integer
	Command register		

Bits	Name	Read	Write	Reset	Description
0…7	Width	✔	✔	x	Widths of 1 to 255 pixels
8…31	Reserved	0	0	x	

Notes: Defines the width of an aliased line. A width 0 is considered to be 1. For aliased lines the LineWidth register holds the desired line width. The range of actual line widths are 1…255; a 0 line width is treated as a line width of 1.

Lines are drawn according to OpenGL rules so wide lines are a sequence of lines offset in X or Y depending on whether the line is X major or Y major. The LineWidthOffset register is normally set to (line width - 1) / 2. For one pixel wide lines the LineWidthOffset is set to 0.

If line stipples are enabled (in the LineMode register) then wide aliased lines will be stippled correctly by repeating the line (offset in X or Y), but with the stipple position re-established for each line used to make up the width.

LineWidthOffset

Name	Type	Offset	Format
LineWidthOffset	Delta	0x94B8	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…7	Offset	✓	✓	x	Offset
8…31	Reserved	0	0	x	

Notes: Defines how far to the left y-major (or lower for x-major) of the true line position an aliased line will be drawn (normally (LineWidth-1)/2). Line width 0 is seen as 1. This sets up the initial offset subtracted from the line's mathematical X (for Y major lines) or Y (for X major lines) before the line is stelled and repeated *LineWidth* times.

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 [25]	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	UseConstantFBWriteData	✔	✔	x	There is no longer any performance advantage to using this bit but it is retained for backwards compatability.
6	BackgroundEnable	✔	✔	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	BackgroundLogicalOp	✔	✔	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.

[25] 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.
13…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
LogicalTexturePageTable Length	Texture *Control register*	0xB4D8	Integer

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 auto-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 [26]	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

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

5…7	FragmentOperation	✔	✔	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	MotionComp8Bits	✔	✔	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: The start index and number of words to fill in the LUT are given by the **LUTTransfer** message with the index in the bits 0…7 and the count in bits 8…13. The **LUTTransfer** message also starts the transfer.
A count of zero loads zero words into the LUT so this effectively disables the loading operation. The count is in multiple of 4 words and the **LUTAddress** is aligned to a 16 byte boundary. The transfer will wrap around in the LUT if necessary.

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 [27]	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 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.
- Readback from Rasterizer unit

[27] The readback for PixelSize comes from the Rasterizer unit.

PointMode
PointModeAnd
PointModeOr

Name	Type	Offset	Format
PointMode PointModeAnd PointModeOr	Delta	0x9490, 0xAAE0, 0xAAE8	bitfield

Control registers

Bits	Name	Read	Write	Reset	Description
0	Antialias Enable	✔	✔	x	Enabled = 1
0	Antialiasing Quality	✔	✔	x	This field defines the quality of antialiased points: 0 4x4 1 8x8

Notes: Defines if and how points are to be antialiased:
AntialiasEnable: enables antialiasing of points, qualified by the *AntialiasEnable* field in the **Begin** register. Note the Point Table must be set up for the corresponding point size (held in **AAPointSize**) and *AntialiasingQuality*.

PointSize

Name	Type	Offset	Format
PointSize	Delta	0x9498	Integer

Control register

Bits	Name	Read	Write	Reset	Description
0…7	Size	✓	✓	x	8 bit integer value from 0 to 255
8…31	Reserved	0	0	x	

Notes: Defines the size of an aliased point. For aliased points the **PointSize** register holds the desired point size. The range of actual point sizes are 1…255; a 0 point size is treated as a point size of 1. Points are drawn according to the OpenGL rules so wide points are drawn as squares centred on the vertex.

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.

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.

Q1Start

Name	Type	Offset	Format
Q1Start	Texture *Control register*	0x8430	Fixed point

Bits	Name	Read	Write	Reset	Description
0…n	Fraction	✓	✓	x	2's complement fixed point fraction
n…31	Integer	✓	✓	x	2's complement fixed point integer

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.

RasterizerMode
RasterizerModeAnd
RasterizerModeOr

Name	Type	Offset	Format
RasterizerMode	Rasterizer	0x80A0	Bitfield
RasterizerModeAnd	Rasterizer	0xABA0	Bitfield
RasterizerModeOr	Rasterizer	0xABA8	Bitfield
	Control register		

Bits	Name	Read [28]	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.

[28] 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	Mask to 0. Reserved for Primitive type - see **Begin** command

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.

ReadMonitorMode
ReadMonitorModeAnd
ReadMonitorModeOr

Name	Type	Offset	Format
ReadMonitorMode	R4 ReadMonitor	0x80F8	Bitfield
ReadMonitorModeAnd	R4 ReadMonitor	0xB5C0	Bitfield
ReadMonitorModeOr	R4 ReadMonitor	0xB5C8	Bitfield
	Control register		

Bits	Name	Read	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…3	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 R4s * StripeHeight. The options are: 0 = 1 4 = 16 1 = 2 5 = 32 2 = 4 6 = 64 3 = 8 7 = 128
4…6	StripeHeight				This field specifies the number of scanlines in a stripe. The options are: 0 = 1 3 = 8 1 = 2 4 = 16 2 = 4.
7	HashFunction				This bit controls the hash function used to index the monitor table from the fragment's xy coordinate. The options are 0 = concatenate the lower 5 bits of x and y together. 1 = xor the lower 10 bits of x and y together.
16…31	StripeOffset				This field should hold the same value as the Rasteriser StripeOffsetY register. The field is a 16 bit 2's complement number.

Notes: Each primitive is assigned a unique number which is recorded in a table for each active pixel the primitive touches. Before the table is updated (during rendering) for a pixel in this primitive it is first tested to see if an earlier primitive had already been assigned to the pixel position. If it has the Suspend Reads mechanism is invoked and the table is reset (i.e. every entry is marked invalid). This significantly reduces unnecessary delays while waiting for a render to complete before starting the next memory writes.

RectangleMode

Name	Type	Offset	Format
RectangleMode	Delta ***Control register***	0x94D0	Float

Bits	Name	Read	Write	Reset	Description
0..31	RectangleMode	✓	✓	x	Holds the data used in the Render command sent to GLINT R4 duringGeomRectangle processing.

Notes: The width and height of the rectangle are held in the **RectangleWidth** and **RectangleHeight** registers as floating point numbers and the **RectangleMode** register holds data passed to the rasteriser in the **Render** command.

RectangleHeight

Name	Type	Offset	Format
RectangleHeight	Delta *Control register*	0x94E0	Float

Bits	Name	Read	Write	Reset	Description
0...31	Height	✓	✓	x	The height of the rectangle as a floating point number

Notes:

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.

RectangleWidth

Name	Type	Offset	Format
RectangleWidth	Delta *Control register*	0x94D8	Float

Bits	Name	Read	Write	Reset	Description
0...31	Width	✓	✓	x	The width of the rectangle as a floating point number

Notes:

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	FastFill Enable	✗	✔	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	Unused	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	✗	✔		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	Reserved	0	0	x	Reserved for primitive type - see **Begin** command

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	2's complement fixed point fraction
n…31	Integer	✓	✓	x	2's complement fixed point integer

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.

ScanLineOwnership

Name	Type	Offset	Format
ScanLineOwnership	Rasterizer ***Control register***	0x80B0	Bitfield

Bits	Name	Read	Write	Reset	Description
0...2	Owner ID	✔	✔	X	
3...5	Stripe ID	✔	✔	X	
6...31	Unused	0	0	X	

Notes: This message defines which R4 owns which scanlines. It has the following fields:
bits 0…2 = mask to isolate the low order bits of Y as these identify the RX which owns this stripe.
bits 3…5 = myId. The stripe this RX owns .
Note that the contents of this register will be different for each R4 in the system.

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 [29]	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.

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

ScreenSize

Name	Type	Offset	Format
ScreenSize	Scissor ***Control register***	0x8198	Bitfield

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.

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

SetDeltaPort

Name	Type	Offset	Format
SetDeltaPort	R4 Delta *Control register*	0x80F0	Bitfield

Bits	Name	Read	Write	Reset	Description
0…30	Port Number	✔	✔	x	Port number
31	Passed through	0	0	x	

Notes: This message sets which Delta port should be made the current port to receive subsequent messages. It is not normally used as an input and is provided only to make testing easier.

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.

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 [30]	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.

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

Stencil

Name	Type	Offset	Format
Stencil	Stencil Command/control register	0x8998	Bitfield

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 [31]	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.

[31] 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 [32]	Write	Reset	Description
0	Unit enable	✔	✔	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.

[32] 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 R4. 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	2's complement fixed point fraction
n…31	Integer	✓	✓	x	2's complement fixed point integer

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 **TextureBaseAddr*n***, to allow multiple updates without duplicating tag data. 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. Readback is via the updated **TextureBaseAddress** register(s).

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 [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 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.

[33] 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 application (or Kd processing) to be added with 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 re 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 *Control register*	0x8500	Integer

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**.

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	✔	✔	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, 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. This is new in Permedia3.
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 = Colour.A 2 = TextureCompositeFactorn.A 3 = Texel0.A 4 = Texel1.A 5 = Texel0.C 6 = Texel1.C where n is the same as the message suffix and 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	

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 [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 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.

[34] 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 = Colour.A 2 = TextureCompositeFactorn.A 3 = Texel0.A 4 = Texel1.A 5 = Texel0.C 6 = Texel1.C where n is the same as the message suffix and 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 = Colour.A 2 = TextureCompositeFactorn.A 3 = Texel0.A 4 = Texel1.A 5 = Texel0.C 6 = Texel1.C where n is the same as the message suffix and 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
TextureCompositeColorMode1	Texture	0xB318	Bitfield
TextureCompositeColorMode1And	Texture	0xB3A0	Bitfield Logic Mask
TextureCompositeColorMode1Or	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 = Colour.A 2 = TextureCompositeFactorn.A 3 = Texel0.A 4 = Texel1.A 5 = Texel0.C 6 = Texel1.C where n is the same as the message suffix and 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 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	InhibitDDA Initialisation	✔	✔	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	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. In MX, Permedia2 and earlier chipsets known as **TextureAddressMode**, but the address data is now part of Texture Read functionality.

TextureData

Name	**Type**	**Offset**	**Format**
TextureData	Localbuffer ***Control register***	0x88E8	Integer

Bits	**Name**	**Read**	**Write**	**Reset**	**Description**
0…31	Texture value	✗	✓	x	32 bit integer value from 0 to 65535

Notes: 32bit raw texture value, formatted to match the format that will be used when the texture is read back from the localbuffer. May include multiple texels (depending on the texel depth), in which case the order of texels within the register will depend on factors such as the byte swap mode, as defined in the TextureFormat register when the texture is subsequently read.

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 [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 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	CombineCaches	✔	✔	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.

[35] 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 [36]	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

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

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	Magnification Filter	✔	✔	x	This field selects the magnification filter to use. The options are 0 = Nearest 1 = Linear
16…18	Minification Filter	✔	✔	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 [37]	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

[37] 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

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 *Control registers*	0xAC38	Bitfield Logic Mask

Bits	Name	Read [38]	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.

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

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.
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 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.
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.
- 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 OpaqueSpan field determines how constant color spans are written (recall the Render command selects between constant color or variable color spans). 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. This bit should be set to 0 for performance reasons when foreground/background processing is not 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.

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 [39]	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 TextureBaseAddr registers upto (and including) the MaxMaxLevel (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.

[39] 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 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.
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.
- 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 OpaqueSpan field determines how constant color spans are written (recall the Render command selects between constant color or variable color spans). 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. This bit should be set to 0 for performance reasons when foreground/background processing is not 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.

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).

TriangleMode
TriangleModeAnd
TriangleModeOr

Name	Type	Offset	Format
TriangleMode	Delta	0x94C8	Float
TriangleModeAnd	Delta	0xAB00	Float
TriangleModeOr	Delta	0xAB08	Float
	Control register		

Bits	Name	Read	Write	Reset	Description
0..31	TriangleMode	✓	✓	x	Defines if and how triangles are antialiased and the message protocols to use when communicating with GLINT.

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.

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.

V0FixedKs
V0FixedKd

Name	Type	Offset	Format
V0FixedKs	R4 Delta	0x9018	Fixed
V0FixedKd	R4 Delta	0x9020	Fixed
	Control registers		

Bits	Name	Read	Write	Reset	Description
0…31		✔	✔	x	Vertex 0 Texture Ks and Kd in fixed point format.

Notes: These registers are deprecated and retained for backward compatibility only.

V0FloatKs
V0FloatKd

Name	Type	Offset	Format
V0FloatKs	R4 Delta	0x9198	Float
V0FloatKd	R4 Delta	0x91A0	Float
	Control registers		

Bits	Name	Read	Write	Reset	Description
0…31		✔	✔	x	Vertex 0 Texture Ks and Kd in floating point format.

Notes: Supported in R4 only

V0FixedR
V0FixedG
V0FixedB
V0FixedA
V0FixedF
V0FixedX
V0FixedY
V0FixedZ

Name	Type	Offset	Format
V0FixedR	R4 Delta	0x9028	Fixed
V0FixedG	R4 Delta	0x9030	Fixed
V0FixedB	R4 Delta	0x9038	Fixed
V0FixedA	R4 Delta	0x9040	Fixed
V0FixedF	R4 Delta	0x9048	Fixed
V0FixedX	R4 Delta	0x9050	Fixed
V0FixedY	R4 Delta	0x9058	Fixed
V0FixedZ	R4 Delta	0x99060	Fixed

Control registers

Bits	Name	Read	Write	Reset	Description
0…31		✔	✔	x	Vertex RGB color, alpha, fog, X, Y and depth

Notes: Vertex 0 color, alpha, fog and coordinate values. These registers are deprecated and retained for backwards compatibility only.

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.

V0FixedS
V0FixedT
V0FixedQ

Name	Type	Offset	Format
V0FixedS	R4 Delta	0x9000	Float
V0FixedT	R4 Delta	0x9008	Float
V0FixedQ	R4 Delta	0x9010	Float

Control registers

Bits	Name	Read	Write	Reset	Description
0…31	Texture	✔	✔	x	Vertex texture value

Notes: The S, T and Q values for vertex 0 as fixed point 2s complement numbers.
The format and clamping range are as defined for the GLINT Gamma - refer to the *GLINT Gamma Programmers Reference Manual* for details. These registers are deprecated and retained for backwards compatibility only.

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.

V1FixedKs
V1FixedKd

Name	Type	Offset	Format
V0FixedKs	R4 Delta	0x9098	Fixed
V0FixedKd	R4 Delta	0x90A0	Fixed
	Control registers		

Bits	Name	Read	Write	Reset	Description
0…31		✔	✔	x	Vertex 1 Texture Ks and Kd in fixed point format.

Notes: These registers are deprecated and retained for backward compatibility only.

V1FloatKs
V1FloatKd

Name	Type	Offset	Format
V1FloatKs	R4 Delta	0x9198	Float
V1FloatKd	R4 Delta	0x91A0	Float
	Control registers		

Bits	Name	Read	Write	Reset	Description
0…31		✔	✔	x	Vertex 1 Texture Ks and Kd in floating point format.

Notes: Supported in R4 only

V1FixedR
V1FixedG
V1FixedB
V1FixedA
V1FixedF
V1FixedX
V1FixedY
V1FixedZ

Name	Type	Offset	Format
V1FixedR	R4 Delta	0x90A8	Fixed
V1FixedG	R4 Delta	0x90B0	Fixed
V1FixedB	R4 Delta	0x90B8	Fixed
V1FixedA	R4 Delta	0x90C0	Fixed
V1FixedF	R4 Delta	0x90C8	Fixed
V1FixedX	R4 Delta	0x90D0	Fixed
V1FixedY	R4 Delta	0x90D8	Fixed
V1FixedZ	R4 Delta	0x90E0	Fixed

Control registers

Bits	Name	Read	Write	Reset	Description
0…31		✔	✔	x	Vertex RGB color, alpha, fog, X, Y and depth

Notes: Vertex 1 color, alpha, fog and coordinate values. These registers are deprecated and retained for backwards compatibility only.

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.

V1FixedS
V1FixedT
V1FixedQ

Name	Type	Offset	Format
V1FixedS	R4 Delta	0x9080	Fixed
V1FixedT	R4 Delta	0x9088	Fixed
V1FixedQs	R4 Delta	0x9090	Fixed
	Control registers		

Bits	Name	Read	Write	Reset	Description
0…31	Texture	✔	✔	x	Vertex texture values

Notes: These registers are deprecated and retained for backwards compatibility only.

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.

V2FixedKs
V2FixedKd

Name	Type	Offset	Format
V2FixedKs	R4 Delta	0x9118	Fixed
V2FixedKd	R4 Delta	0x9120	Fixed

Control registers

Bits	Name	Read	Write	Reset	Description
0…31		✔	✔	x	Vertex 1 Texture Ks and Kd in fixed point format.

Notes: These registers are deprecated and retained for backward compatibility only.

V2FloatKs
V2FloatKd

Name	Type	Offset	Format
V2FloatKs	R4 Delta	0x9298	Float
V2FloatKd	R4 Delta	0x92A0	Float

Control registers

Bits	Name	Read	Write	Reset	Description
0…31		✔	✔	x	Vertex 2 Texture Ks and Kd in floating point format.

Notes: Supported in R4 only

V2FixedR
V2FixedG
V2FixedB
V2FixedA
V2FixedF
V2FixedX
V2FixedY
V2FixedZ

Name	Type	Offset	Format
V2FixedR	R4 Delta	0x9128	Fixed
V2FixedG	R4 Delta	0x9130	Fixed
V2FixedB	R4 Delta	0x9138	Fixed
V2FixedA	R4 Delta	0x9140	Fixed
V2FixedF	R4 Delta	0x9148	Fixed
V2FixedX	R4 Delta	0x9150	Fixed
V2FixedY	R4 Delta	0x9158	Fixed
V2FixedZ	R4 Delta	0x9160	Fixed

Control registers

Bits	Name	Read	Write	Reset	Description
0…31		✔	✔	x	Vertex RGB color, alpha, fog, X, Y and depth

Notes: Vertex 2 color, alpha, fog and coordinate values. These registers are deprecated and retained for backwards compatibility only.

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.

V2FixedS
V2FixedT
V2FixedQ

Name	Type	Offset	Format
V2FixedS	R4 Delta	0x9100	Fixed
V2FixedT	R4 Delta	0x9108	Fixed
V2FixedQ	R4 Delta	0x9110	Fixed
	Control registers		

Bits	Name	Read	Write	Reset	Description
0…31	Texture	✔	✔	x	Vertex texture values

Notes: V2 vertex texture values. These registers are deprecated and included for backwards compatibility only.

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.

Vertex0

Name	Type	Offset	Format
Vertex0	Input	0xB7B8	Integer
	Control register		

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
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
WindowAnd
WindowOr

Name	Type	Offset	Format
Window	Localbuffer	0x8980	Bitfield
WindowAnd	Localbuffer	0xAB80	Bitfield
WindowOr	Localbuffer *Control register*	0xAB88	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	This field holds the current frame count used as part of the Fast Clear Planes (FCP) mechanism
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 (if enabled) added to the vertices' X coordinate just before rasterization. This ensures uniform floating point precision across the full range of screen coordinates. It can be easily removed for absolute coordinate addressing.

YBias

Name	Type	Offset	Format
YBias	Delta *Control register*	0x9488	Float

Bits	Name	Read	Write	Reset	Description
0…31	Offset	✓	✓	x	

Notes: This register holds the single precision floating point bias (if enabled) added to the vertices' Y coordinate just before rasterization. This ensures uniform floating point precision across the full range of screen coordinates. It can be easily removed for absolute coordinate addressing.

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.

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. Not all legacy registers are included.

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	✕
AntialiasModeAnd	✕	✔	Alpha test	AC00	x	bitfield	✕

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
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	✕
Begin	✕	✔	Delta	9590	x	bitfield	✕
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	
ByDMAReadCommandBase	✔	✔	Bypass Control	0378		Integer	

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
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	✔	✔	Color DDA	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	✗
DeltaMode	✔	✔	Delta	9300	x	bitfield	✗

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
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	✗
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	✗
DMARectangleWrite	✗	✔	Host In	A9C8	x	bitfield	✗
DMARectangleWriteAddress	✔	✔	Host In	A9D0	x	integer	✗
DMARectangleWriteLinePitch	✔	✔	Host In	A9D8	x	integer	✗
DownloadAddress	✔	✔		B0d0	x	integer	✗
DownloadData	✔	✔		B0d8	x	integer	✗

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
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	Fixed	✕
DrawLine0	✕	✔	Delta	9318	x	fixed	✔
DrawLine1	✕	✔	Delta	9320	x	fixed	✔
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	✕
dXDom	✔	✕	Raster-izer	8008	x	fixed	✕
dXSub	✔	✕	Raster-izer	8018	x	fixed	✕
dY	✔	✕	Raster-izer	8028	x	fixed	✕
dZdxL	✔	✔	Depth & Fog	89C8	x	Fixed	✕

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
dZdxU	✔	✔	Depth & Fog	89C0	x	Fixed	✕
dZdyDomL	✔	✔	Depth & Fog	89D8	x	Bitfield	✕
dZdyDomU	✔	✔	Depth & Fog	89D0	x	Fixed	✕
End	✕	✔	Delta	9598	x	fixed	✕
EndOfFeedback	✔	✔	Host Out	8FF8	x	tag	✕
ErrorFlags			Control Status	0038		bitfield	
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	✔	✕	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	✕
FBWriteModeOr	✕	✔	FB Write	ACF8	x	bitfield	✕
FifoControl	✔	✔	Video Control	3078		bitfield	
FIFODiscon	✔	✔	Control Status	0068		bitfield	
FillBackgroundColor	✕	✔	2D Set Up	8330	x	integer	✕

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
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	✕
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	✕

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
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	Integer	✗
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	
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	✗

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
KdRStart	✔	✔	Texture	8D00	x	fixed	✕
KsBStart	✔	✔	Texture Application	8CB0	x	fixed	✕
KsGStart	✔	✔	Texture Application	8C98	x	fixed	✕
KsRStart	✔	✔	Texture Application	8C80	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	✕
LBReadModeAnd	✕	✔	Delta	AB90	x	bitfield	✕
LBReadModeOr	✕	✔	Delta	AB98	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	✕
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	✕

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
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	✕
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	

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
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	✔
PointMode	✔	✔	Delta	9490	x	bitfield	✕
PointModeAnd	✕	✔	Delta	AAE0	x	bitfield	✕
PointModeOr	✕	✔	Delta	AAE8	x	bitfield	✕
PointSize	✔	✔	Delta	9498	x	integer	✕
PointTable[0…3]	✔	✔	Raster-izer	8080	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	✕
RasterizerModeOr	✕	✔	Raster-izer	ABA8	x	bitfield	✕
RDIndexControl	✔	✔	RAMDAC Control	4038		Integer	

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
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	
ReadMonitorMode	✔	✔	Delta	80F8	x	Bitfield	✕
ReadMonitorModeAnd	✕	✔	Delta	B5C0	x	bitfield	✕
ReadMonitorModeOr	✕	✔	Delta	B5C8	x	bitfield	✕
RectangleHeight	✔	✔	Delta	94E0	x	float	✕
RectangleMode	✔	✔	Delta	94D0	x	bitfield	✕
RectanglePosition	✔	✔	2D Set Up	B600	x	integer	✕
RectangleWidth	✔	✔	Delta	94D8	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	✕

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
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	✔
ScanLineOwnership	✔	✔	Raster-izer	80B0	x	Bitfield	✕
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	✕
SetDeltaPort	✕	✔	Delta	80F0	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	✕

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
StencilMode	✔	✔	Stencil	8988	x	Bitfield	✕
StencilModeAnd	✕	✔	Stencil	AC60	x	bitfield	✕
StencilModeOr	✕	✔	Stencil	AC68	x	bitfield	✕
StripeOffsetY	✔	✔	Raster-izer	80C8	x	fixed	✕
SuspendUntilFrameBlank	✕	✔	Framebu ffer 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	
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	✕

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
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	✗
TextureChromaLower0	✔	✔	Color DDA	84F0	x	Bitfield	✗
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	✗

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
TextureCompositeColorMode1And	✗	✔	Texture Composite	B3A0	x	bitfield	✗
TextureCompositeColorMode1Or	✗	✔	Texture Composite	B3A8	x	bitfield	✗
TextureCompositeFactor0	✔	✔	Texture Composite	B328	X	bitfield	
TextureCompositeFactor1	✔	✔	Texture Composite	B330	X	bitfield	✗
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	✔	✗	Control Status	0108		bitfield	
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	✗
TextureMapSize	✔	✔	Texture Read	B428	x	integer	✗
TextureMapWidth[16]	✔	✔	Texture Read	8580	x	Bitfield	✗

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
TextureOperation	✔	✕	Control Status	0110		integer	
TextureReadMode0	✔	✔	Texture Read	B400	x	bitfield	✕
TextureReadMode0And	✕	✔	Texture Read	AC30	x	bitfield	✕
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	✔
TriangleMode	✔	✔	Delta	94C8	x	bitfield	✕
TriangleModeAnd	✕	✔	Delta	AB10	x	bitfield	✕
TriangleModeOr	✕	✔	Delta	AB18	x	bitfield	✕
TStart	✔	✔	Texture coord	83A0	x	Fixed	✕
UpdateLineStippleCounters	✕	✔	Stipple	81B8	x	Bitfield	✔
UpdateLogicalTextureInfo	✕	✔	Texture Read	B368	X	tag	✔
V0FixedF	✔	✔	Delta	9048	x	fixed	✕
V0FixedQ	✔	✔	Delta	9010	x	fixed	✕
V0FixedS	✔	✔	Delta	9000	x	fixed	✕
V0FixedT	✔	✔	Delta	9008	x	fixed	✕
V0FixedX	✔	✔	Delta	9050	x	fixed	✕
V0FixedY	✔	✔	Delta	9058	x	fixed	✕
V0FixedZ	✔	✔	Delta	9060	x	fixed	✕
V0FloatA	✔	✔	Delta	91C0	x	float	✕
V0FloatB	✔	✔	Delta	91B8	x	float	✕
V0FloatF	✔	✔	Delta	91C8	x	float	✕
V0FloatG	✔	✔	Delta	91B0	x	float	✕
V0FloatKd	✔	✔	Delta	91A0	x	float	✕
V0FloatKd	✔	✔	Delta	92A0	x	float	✕
V0FloatKs	✔	✔	Delta	9198	x	float	✕
V0FloatKs	✔	✔	Delta	9298	x	float	✕
V0FloatQ	✔	✔	Delta	9010	x	float	✕
V0FloatR	✔	✔	Delta	91A8	x	float	✕
V0FloatS	✔	✔	Delta	9000	x	float	✕
V0FloatT	✔	✔	Delta	9008	x	float	✕
V0FloatX	✔	✔	Delta	91D0	x	float	✕
V0FloatY	✔	✔	Delta	91D8	x	float	✕
V0FloatZ	✔	✔	Delta	91E0	x	float	✕
V1FixedF	✔	✔	Delta	90C8	x	fixed	✕

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
V1FixedQ	✔	✔	Delta	9090	x	float	✕
V1FixedS	✔	✔	Delta	9080	x	float	✕
V1FixedT	✔	✔	Delta	9088	x	float	✕
V1FixedX	✔	✔	Delta	90D0	x	fixed	✕
V1FixedY	✔	✔	Delta	90D8	x	fixed	✕
V1FixedZ	✔	✔	Delta	90E0	x	fixed	✕
V1FloatA	✔	✔	Delta	9240	x	float	✕
V1FloatB	✔	✔	Delta	9238	x	float	✕
V1FloatF	✔	✔	Delta	9248	x	float	✕
V1FloatG	✔	✔	Delta	9230	x	float	✕
V1FloatKd	✔	✔	Delta	9220	x	float	✕
V1FloatKs	✔	✔	Delta	9218	x	float	✕
V1FloatQ	✔	✔	Delta	9210	x	float	✕
V1FloatR	✔	✔	Delta	9228	x	float	✕
V1FloatS	✔	✔	Delta	9200	x	float	✕
V1FloatT	✔	✔	Delta	9208	x	float	✕
V1FloatX	✔	✔	Delta	9250	x	float	✕
V1FloatY	✔	✔	Delta	9258	x	float	✕
V1FloatZ	✔	✔	Delta	9260	x	float	✕
V2FixedF	✔	✔	Delta	9148	x	fixed	✕
V2FixedQ	✔	✔	Delta	9110	x	float	✕
V2FixedS	✔	✔	Delta	9100	x	float	✕
V2FixedT	✔	✔	Delta	9108	x	float	✕
V2FixedX	✔	✔	Delta	9150	x	fixed	✕
V2FixedY	✔	✔	Delta	9158	x	fixed	✕
V2FixedZ	✔	✔	Delta	9160	x	fixed	✕
V2FloatA	✔	✔	Delta	92C0	x	float	✕
V2FloatB	✔	✔	Delta	92B8	x	float	✕
V2FloatF	✔	✔	Delta	92C8	x	float	✕
V2FloatG	✔	✔	Delta	92B0	x	float	✕
V2FloatQ	✔	✔	Delta	9290	x	float	✕
V2FloatR	✔	✔	Delta	92A8	x	float	✕
V2FloatS	✔	✔	Delta	9280	x	float	✕
V2FloatT	✔	✔	Delta	9288	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	✕
VertexBaseAddress	✔	✔	Host In	B708	x	integer	✕
VertexControl	✔	✔	Host In	B798	x	bitfield	✕

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
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	
VideoOverlayShrinkXDelta	✔	✔	Video Overlay Control	3158		bitfield	

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
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	
VSDMACommandBase	✔	✔	Video Stream Control	5AC8		integer	
VSDMACommandCount	✔	✔	Video Stream Control	5AD0		integer	
VSDMAMode	✔	✔	Video Stream Control	5AC0		bitfield	
VsEnd	✔	✔	Video Control	3050		integer	
VSSerialBusControl	✔	|	Video Stream Control	5810		bitfield	
VsStart	✔	✔	Video Control	3048		integer	
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	✕

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
YLimits	✔	✔	Raster-izer	80A8	x	Bitfield	✕
YUVMode	✔	✔	YUV Unit	8F00	X	bitfield	✕
ZfogBias	✔	✔	Fog	86B8	x	Float	✕
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	

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
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	✔	✕	Control Status	0108		bitfield	
TextureOperation	✔	✕	Control Status	0110		integer	
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	

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
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	
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	

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
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	
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	

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
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	
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	
VSStatus	✔	✕	Video Stream Control	5808		bitfield	

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
VSSerialBusControl	✔	|	Video Stream Control	5810		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	✕
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	✕
ScanLineOwnership	✔	✔	Raster-izer	80B0	x	Bitfield	✕
WaitForCompletion	✕	✔	Raster-izer	80B8	x	Bitfield	✔

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
PixelSize	✔	✔	Raster-izer	80C0	x	Bitfield	✔
StripeOffsetY	✔	✔	Raster-izer	80C8	x	fixed	✕
SetDeltaPort	✕	✔	Delta	80F0	x	Bitfield	✔
ReadMonitorMode	✔	✔	Delta	80F8	x	Bitfield	✕
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	✕
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	✕

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
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	✕
dQdyDom	✔	✔	Texture coord	83C8	x	Fixed	✕
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	✕

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
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	✕
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	✕
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	✔

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
TextTGlyphAddr4	✕	✔	Raster-izer	8740	x	Integer	✕
TextRender2DGlyph4	✕	✔	Raster-izer	8748	x	Bitfield	✔
TextTGlyphAddr5	✕	✔	Raster-izer	8750	x	Integer	✕
TextRender2DGlyph5	✕	✔	Raster-izer	8758	x	Bitfield	✔
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	✔	✔	Color DDA	87F0	x	Bitfield	✕

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
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	✕
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	✕
HostInID	✔	✔	Host In	8900	x	Integer	✕
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	✕
FBColor	✔	✕	FB Write	8A98	x	n/a	✕
FBWriteMode	✔	✔	FB Write	8AB8	x	bitfield	✕
FBHardwareWriteMask	✔	✔	FB Write	8AC0	x	mask	✕

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
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	✔
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	✕

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
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	✗
EndOfFeedback	✔	✔	Host Out	8FF8	x	tag	✗
V0FixedS	✔	✔	Delta	9000	x	fixed	✗
V0FixedT	✔	✔	Delta	9008	x	fixed	✗
V0FixedQ	✔	✔	Delta	9010	x	fixed	✗
V0FixedF	✔	✔	Delta	9048	x	fixed	✗
V0FixedX	✔	✔	Delta	9050	x	fixed	✗
V0FixedY	✔	✔	Delta	9058	x	fixed	✗
V0FixedZ	✔	✔	Delta	9060	x	fixed	✗
V1FixedS	✔	✔	Delta	9080	x	float	✗
V1FixedT	✔	✔	Delta	9088	x	float	✗
V1FixedQ	✔	✔	Delta	9090	x	float	✗
V1FixedF	✔	✔	Delta	90C8	x	fixed	✗
V1FixedX	✔	✔	Delta	90D0	x	fixed	✗
V1FixedY	✔	✔	Delta	90D8	x	fixed	✗
V1FixedZ	✔	✔	Delta	90E0	x	fixed	✗
V2FixedS	✔	✔	Delta	9100	x	float	✗
V2FixedT	✔	✔	Delta	9108	x	float	✗
V2FixedQ	✔	✔	Delta	9110	x	float	✗
V2FixedF	✔	✔	Delta	9148	x	fixed	✗
V2FixedX	✔	✔	Delta	9150	x	fixed	✗
V2FixedY	✔	✔	Delta	9158	x	fixed	✗
V2FixedZ	✔	✔	Delta	9160	x	fixed	✗
V0FloatS	✔	✔	Delta	9000	x	float	✗
V0FloatT	✔	✔	Delta	9008	x	float	✗
V0FloatQ	✔	✔	Delta	9010	x	float	✗
V0FloatKs	✔	✔	Delta	9198	x	float	✗
V0FloatKd	✔	✔	Delta	91A0	x	float	✗
V0FloatR	✔	✔	Delta	91A8	x	float	✗

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
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	✕
V1FloatS	✔	✔	Delta	9200	x	float	✕
V1FloatT	✔	✔	Delta	9208	x	float	✕
V1FloatQ	✔	✔	Delta	9210	x	float	✕
V1FloatKs	✔	✔	Delta	9218	x	float	✕
V1FloatKd	✔	✔	Delta	9220	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	✕
V2FloatS	✔	✔	Delta	9280	x	float	✕
V2FloatT	✔	✔	Delta	9288	x	float	✕
V2FloatQ	✔	✔	Delta	9290	x	float	✕
V0FloatKs	✔	✔	Delta	9298	x	float	✕
V0FloatKd	✔	✔	Delta	92A0	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	✕
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	✔
XBias	✔	✔	Delta	9480	x	float	✕
YBias	✔	✔	Delta	9488	x	float	✕
PointMode	✔	✔	Delta	9490	x	bitfield	✕
PointSize	✔	✔	Delta	9498	x	integer	✕
AAPointSize	✔	✔	Delta	94A0	x	float	✕
LineMode	✔	✔	Delta	94A8	x	bitfield	✕

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
LineWidth	✔	✔	Delta	94B0	x	integer	✕
LineWidthOffset	✔	✔	Delta	94B8	x	integer	✕
AALineWidth	✔	✔	Delta	94C0	x	float	✕
TriangleMode	✔	✔	Delta	94C8	x	bitfield	✕
RectangleMode	✔	✔	Delta	94D0	x	bitfield	✕
RectangleWidth	✔	✔	Delta	94D8	x	integer	✕
RectangleHeight	✔	✔	Delta	94E0	x	float	✕
Begin	✕	✔	Delta	9590	x	bitfield	✕
End	✕	✔	Delta	9598	x	fixed	✕
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	✕
DMARectangleWriteAddress	✔	✔	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	✕
PointModeAnd	✕	✔	Delta	AAE0	x	bitfield	✕
PointModeOr	✕	✔	Delta	AAE8	x	bitfield	✕
LineModeAnd	✕	✔	Delta	AAF0	x	bitfield	✕
LineModeOr	✕	✔	Delta	AAF8	x	bitfield	✕
TriangleModeAnd	✕	✔	Delta	AB10	x	bitfield	✕
TriangleModeOr	✕	✔	Delta	AB18	x	bitfield	✕
WindowAnd	✕	✔	Stencil	AB80	x	bitfield	✕
WindowOr	✕	✔	Stencil	AB88	x	bitfield	✕
LBReadModeAnd	✕	✔	Delta	AB90	x	bitfield	✕
LBReadModeOr	✕	✔	Delta	AB98	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	✕

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
ColorDDAModeAnd	✗	✔	Color DDA	ABE0	x	bitfield	✗
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 Application	AC50	x	bitfield	✗
TextureApplication ModeOr	✗	✔	Texture Application	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	✗

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
ChromaTestModeAnd	✗	✔	Color DDA & Alpha Blend	ACC0	x	bitfield	✗
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	✗
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	✗

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
AlphaSourceColor	✔	✔	Alpha blend	AF80	x	integer	✕
AlphaDestColor	✔	✔	Alpha blend	AF88	x	bitfield	✕
ChromaPassColor	✔	✔	Color DDA & Alpha Blend	AF90	x	bitfield	✕
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	✕
DownloadAddress	✔	✔		B0d0	x	integer	✕
DownloadData	✔	✔		B0d8	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	✕

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
TextureCompositeColorMode1	✔	✔	Texture Composite	B318	X	bitfield	✕
TextureCompositeAlphaMode1	✔	✔	Texture Composite	B320	X		✕
TextureCompositeFactor0	✔	✔	Texture Composite	B328	X	bitfield	
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	✕

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
TextureCompositeAlphaMode1Or	✗	✔	Texture Composite	B3B8	x	bitfield	✗
TextureIndexMode0And	✗	✔	Texture Index	B3C0	x	bitfield	✗
TextureIndexMode0Or	✗	✔	Texture Index	B3C8	x	bitfield	✗
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	✗

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
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	✗
ReadMonitorModeAnd	✗	✔	Delta	B5C0	x	bitfield	✗
ReadMonitorModeOr	✗	✔	Delta	B5C8	x	bitfield	✗
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	✗

Name	Read back	Write	Unit Name	Offset	Reset Value	Format	Com-mand
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	✗
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	✗

INDEX