GLINT R5®

Reference Guide Volume III -

Graphics Registers

PROPRIETARY AND CONFIDENTIAL INFORMATION

3D*labs*®

GLINT R5®

Reference Guide Volume III -

Graphics Registers

PROPRIETARY AND CONFIDENTIAL INFORMATION

Issue 4

Proprietary Notice

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* GmbH
Breckenheimer Weg 29
65205 Wiesbaden
Deutschland
Tel: +49 6122 916 778
Fax: +49 6122 919 646

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: +1 (408) 530-4700
Fax: +1 (408) 530-4701

Change History

Document	Issue	Date	Change
172.1.3	1	15/02/2000	Creation
172.1.3	2	03/05/2000	Major update to preliminary release level
172.1.3	3	25/07/2000	Removed *Arrayindexmultiple, Arrayindexoffset, Multiindexunits* bit in **ArrayMode**,180500; complete register integration 090600; add more from Gamma 2/3, delete **DMAReadGlintSource**, 130600.
172.1.3	4	05/03/2001	Deleted duplicate fields adjacent to VertexMachine Mode, deleted duplicate ViewPortScaleX, corrected Color register offset, added TransformMode register, corrected font for ticks, added bookmarks, added TransformCurrent register, clarified OpaqueSpan bit, minor font and format changes

Table of Contents

5

Graphics Registers

This chapter lists graphics core 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.
Shaded heading	On a register indicates that the register is part of the Gamma pipeline. Unshaded indicates that the register is part of the GLINT rasterizer pipeline.

The core registers are of two general types: geometry acceleration and rasterization. Geometry registers are shown with a shaded title bar, rasterization registers are not. For a description of the units in each part of the pipeline, see "Pipeline Architecture" in *Reference Guide* Volume I

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 Gamma and GLINT family may define a meaning for the unused values.

AALineExtend

Name	Type	Offset	Format
AALineExtend	Cull *Gamma Control register* Broadcast	0xC5A8	Int

Bits	Name	Read	Write	Reset	Description
0..31	Extent	✓	✓	x	8.3 unsigned fixed point

Notes: Holds the number of scanlines to grow the y extent by when dealing with antialiased lines. It is updated whenever the **AALineWidth** value is updated. The width is rounded up to the nearest 0.5 (4x4) or 0.25 (8x8) scanline, divided by two and converted to 8.3 fixed point format and loaded into the **AALineExtend** register.

AAALineWidth

Name	Type	Offset	Format
AALineWidth	Delta *GLINT 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 not drawn and lines with a negative width draw a line of the same positive width . The width is in pixels. Fractional width (>1.0) are drawn but may not be continuous due to sampling errors. Antialiased lines are drawn as rectangles aligned to the direction of the line.

AAPointExtend

Name	Type	Offset	Format
AAPointExtend	Cull *Gamma Control register* Broadcast	0xC598	Int

Bits	Name	Read	Write	Reset	Description
0..31	Extent	✓	✓	x	8.3 unsigned fixed point

Notes: Holds the number of scanlines to grow the y extent by when dealing with antialiased points. It is updated whenever the **AAPointWidth** value in the Delta Unit is updated.
Single pixel points should have the PointExtend register set to 0. Wide points should have their integer width rounded up to the nearest even width and half this value stored in the PointExtend register (left shifting by 3 to fix the binary point in the correct place).

AAPointSize

Name	Type	Offset	Format
AAPointSize	Delta	0x94A0	Float
	GLINT Control register		

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

Notes: Defines the diameter of antialiased points. In theory any size antialiased point can be defined but in fact the Point Table restricts the diameter to 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}

AATriangleExtend

Name	Type	Offset	Format
AATriangleExtend	Cull	0xC5B8	Int
	Control		
	Broadcast		

Bits	Name	Read	Write	Reset	Description
0..31	Extent	✓	✓	x	8.3 unsigned fixed point

Notes: Holds the number of scanlines to grow the y extent by when dealing with antialiased triangles. It is normally set to 0.

AlphaBlendAlphaMode
AlphaBlendAlphaModeAnd
AlphaBlendAlphaModeOr

Name	Type	Offset	Format
AlphaBlendAlphaMode	Alpha Blend	0xAFA8	Bitfield
AlphaBlendAlphaModeAnd	Alpha Blend	0xAD30	Bitfield Logic Mask
AlphaBlendAlphaModeOr	Alpha Blend	0xAD38	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 blend interpolations etc. refer to the *GLINT R5 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

	Format	Color Order	Name	Internal Color Channels 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	0	0	0

AlphaBlendColorMode
AlphaBlendColorModeAnd
AlphaBlendColorModeOr

Name	Type	Offset	Format
AlphaBlendColorMode	Alpha Blend	0xAFA0	Bitfield
AlphaBlendColorModeAnd	Alpha Blend	0xACB0	Bitfield Logic Mask
AlphaBlendColorModeOr	Alpha Blend	0xACB8	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.
25	PixelHistory Enable	✔	✔	x	This bit, when set, keeps track of the 8 most recent pixels (from active steps) and uses the color read from the history list instead of the active step. In conjunction with the ReadMonitor unit it avoids frequent use of the Suspend Reads mechanism and is particularly useful when drawing antialiased lines. Do not enable if the calculated color in this unit will not match that written to memory, e.g. logical ops or write masking. During Chroma Test the pixel history list is automatically disabled..

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	0x8800	Bitfield
AlphaTestModeAnd	AlphaBlend	0xABF0	Bitfield Logic Mask
AlphaTestModeOr	AlphaBlend	0xABF8	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	0x8808	Bitfield
AntialiasModeAnd	Alpha Test	0xABF0	Bitfield Logic Mask
AntialiasModeOr	Alpha Test	0xABF8	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.

ArrayControl
ArrayControlAnd
ArrayControlOr

Name	Type	Offset	Format
ArrayCor trol	Vertex Array	0xCD80	Bitfield
ArrayCor trolAnd	Vertex Array	0xCD90	Bitfield Logic Mask
ArrayCor trolOr	Vertex Array	0xCDA0	Bitfield Logic Mask
	Control register		

Bits	Name	Read	Write	Reset	Description
0	IndexProtocol	✓	✓	x	0 = PCI 1 = AGP
1	Alignment	✓	✓	x	0 = Off 1 = align to 64 bit boundary where possible
2, 3	ByteSwap	✓	✓	x	0 = ABCD (no swap) 1 = BADC 2 = CDAB 3 = DCBA
4..6	BurstSize	✓	✓	x	Log2 of size of DMA burst in DWords
7	DataProtocol	✓	✓	x	0 = PCI, 1 = AGP
8..31	Reserved	✓	✗	0	

Notes:

ArrayDataAddress[0..15]

Name	Type	Offset	Format
ArrayInd xAddress[0..15]	Vertex Array	0xD200	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0..1	reserved	✓	✗	0	
2..31	Address	✓	✓	X	32 bit aligned address of base of buffer

Notes:

ArrayDataID

Name	Type	Offset	Format
ArrayDat aID	Vertex Array	0xD480	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
16..31	Reserved	✓	✗	0	

Notes:

ArrayDataIndex

Name	Type	Offset	Format
ArrayDat\Index	Vertex Array *Control register*	0xD380	Bitfield

Bits	Name	Read	Write	Reset	Description
0..15	Mask	✓	✓	X	Each bit enables a data array for access by the corresponding Index command
16..31	Reserved	✓	✗	0	

Notes:

ArrayDataLoad

Name	Type	Offset	Format
ArrayDat\Load	Vertex Array *Control register*	0xCE38	Integer

Bits	Name	Read	Write	Reset	Description
0..31	Offset	✓	✓	X	Offset into array of data to read

Notes:

ArrayDataMultiple

Name	Type	Offset	Format
ArrayDat\Multiple	Vertex Array *Control register*	0xCE18	Bitfield

Bits	Name	Read	Write	Reset	Description
0..29	Count	✓	✓	X	Number of data items to read from array
30, 31	Reserved	✓	✗	0	

Notes:

ArrayDataOffset

Name	Type	Offset	Format
ArrayDat :Offset	Vertex Array *Control register*	0xCE08	Integer

Bits	Name	Read	Write	Reset	Description
0..31	Offset	✓	✓	X	First data to use with ArrayDataMultiple

Notes:

ArrayDataSingle

Name	Type	Offset	Format
ArrayDat :Single	Vertex Array *Control register*	0xCE10	Integer

Bits	Name	Read	Write	Reset	Description
0..31	Offset	✓	✓	X	Offset into array of index pointing to data

Notes:

ArrayDataSize[0..15]

Name	Type	Offset	Format
ArrayDat :Size[0..15]	Vertex Array *Control register*	0xD280	Bitfield

Bits	Name	Read	Write	Reset	Description
0, 1	reserved	✓	✗	0	
2..15	Size	✓	✓	X	Number of 32 bit words to be read
16, 17	Reserved	✓	✗	0	
18..31	Stride	✓	✓	X	Number of 32 bit words between consecutive indices

Notes:

ArrayEnable
ArrayEnableAnd
ArrayEnableOr

Name	Type	Offset	Format
ArrayEnable	Vertex Array	0xCD88	Bitfield
ArrayEnableAnd	Vertex Array	0xCD98	Bitfield Logic Mask
ArrayEnableOr	Vertex Array	0xCDA8	Bitfield Logic Mask
	Control register		

Bits	Name	Read	Write	Reset	Description
0..15	Index[0..15]	✓	✓	x	Enable read of index buffer[0..15]
16..31	Data[0..15]	✓	✓	x	Enable read of data buffer [0..15]

Notes: The number of index and data buffers supported will vary with the implementation. Setting the enable bit has no effect if the buffer is not present in the chip. The register reads back as set whether the buffer exists or not.

ArrayFormatField[0..3]
ArrayFormatField[0..3]And
ArrayFormatField[0..3]Or

Name	Type	Offset	Format
ArrayFormatField	Vertex Array	0xCBE0	Integer
ArrayFormatFieldAnd	Vertex Array	0xCC60	Integer Logic Mask
ArrayFormatFieldOr	Vertex Array	0xCCE0	Integer Logic Mask
	Control register		

Bits	Name	Read	Write	Reset	Description
0, 1	1	✓	✓	x	32 bit word in ArrayTagTable entry to load.
2, 3	2	✓	✓	X	32 bit word in ArrayTagTable entry to load.
4…29	3…14	✓	✓	X	Further 32 bit words in ArrayTagTable entry to load.
30, 31	15	✓	✓	X	Final 32 bit word in ArrayTagTable entry to load

Notes:

ArrayFormatLoad[0..1]
ArrayFormatLoad[0..1]And
ArrayFormatLoad[0..1]Or

Name	Type	Offset	Format
ArrayFor natLoad	Vertex Array	0xCB80	Integer
ArrayFor natLoadAnd	Vertex Array	0xCC00	Integer Logic Mask
ArrayFor natLoadOr	Vertex Array	0xCC80	Integer Logic Mask
	Control register		

Bits	Name	Read	Write	Reset	Description
0..31	Load	✓	✓	x	Mask of 32 bit words loaded for each vertex

Notes:

ArrayFormatRef[0..7]
ArrayFormatRef[0..7]And
ArrayFormatRef[0..7]Or

Name	Type	Offset	Format
ArrayFor natRef[0..7]	Vertex Array	0xCBA0	Integer
ArrayFor natRef[0..7]And	Vertex Array	0xCC20	Integer Logic Mask
ArrayFor natRef[0..7]Or	Vertex Array	0xCCA0	Integer Logic Mask
	Control register		

Bits	Name	Read	Write	Reset	Description
0..3	0	✓	✓	x	Entry in ArrayTagTable to load
4..7	1	✓	✓	x	Entry in ArrayTagTable to load
08..11	2	✓	✓	x	Entry in ArrayTagTable to load
12..15	3	✓	✓	x	Entry in ArrayTagTable to load
16..19	4	✓	✓	x	Entry in ArrayTagTable to load
20..23	5	✓	✓	x	Entry in ArrayTagTable to load
24..27	6	✓	✓	x	Entry in ArrayTagTable to load
28..31	7	✓	✓	x	Entry in ArrayTagTable to load

Notes:

ArrayFormatValid[0..1]
ArrayFormatValid[0..1]And
ArrayFormatValid[0..1]Or

Name	Type	Offset	Format
ArrayFor natValid	Vertex Array	0xCB90	Integer
ArrayFor natValidAnd	Vertex Array	0xCC10	Integer Logic Mask
ArrayFor natValidOr	Vertex Array	0xCC90	Integer Logic Mask
	Control register		

Bits	Name	Read	Write	Reset	Description
0..31	Valid	✓	✓	x	Mask of 32 bit words valid for each vertex

Notes:

ArrayIndex[0…15]

Name	Type	Offset	Format
ArrayInd x[0]	Vertex Array	0xD300	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0..29	Index	✓	✓	x	Index to use to reference data array
30, 31	Shift	✓	✓	X	Private data field

Notes:

ArrayIndexAddress[0…15]

Name	Type	Offset	Format
ArrayInd xAddress[0..15]	Vertex Array	0xD100	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0..31	Address	✓	✓	x	Byte address of base of buffer

Notes:

ArrayIndexMultiple

Name	Type	Offset	Format
ArrayInd xMultiple	Vertex Array *Control register*	0xCE28	Integer

Bits	Name	Read	Write	Reset	Description
0..31	Count	✗	✓	x	Number of indices to read from array.

Notes:

ArrayIndexOffset

Name	Type	Offset	Format
ArrayInd xOffset	Vertex Array *Control register*	0xCE00	Integer

Bits	Name	Read	Write	Reset	Description
0..31	Offset	✓	✓	x	First index to use with ArrayIndexMultiple

Notes:

ArrayIndexSingle

Name	Type	Offset	Format
ArrayInd xSingle	Vertex Array *Control register*	0xCE20	Integer

Bits	Name	Read	Write	Reset	Description
0..31	Offset	✓	✓	x	Offset into array of index pointing to data

Notes:

ArrayIndexSize

Name	Type	Offset	Format
ArrayIndexSize	Vertex Array *Control register*	0xD180	Bitfield

Bits	Name	Read	Write	Reset	Description
0..2	Size	✓	✓	X	Valid entries: 1, 2, 4
3..15	Reserved	✓	✗	X	
16..18	Stride	✓	✓	X	Valid entries: 1, 2, 4
19..31	Reserved	✓	✗	0	

Notes:

ArrayMode
ArrayModeAnd
ArrayModeOr

Name	Type	Offset	Format
ArrayMode	Command	0xB0	Bitfield
ArrayModeAnd	Command	0xB8	Bitfield Logic Mask
ArrayModeOr	Command *Control register*	0xC0	Bitfield Logic Mask

Bits	Name	Read	Write	Reset	Description
0	IndexCache Enable	✓	✓	x	0 = Cache Off 1 = Cache On
1	Reserved	✗	✗	x	Reserved for multiple index array support, must be cleared normally
2	MultiDataUnits	✓	✓	x	0 = off 1 = more than one data unit in use
3	D3DProvoking Vertex	✓	✓	x	0 = OpenGL rules 1 = D3D rules
4..31	Reserved	✓	✗	0	

Notes: Bit 1 may cause a lockup if Set. It should be Reset to 0.

ArrayTagTable[0..15]

Name	Type	Offset	Format
ArrayTag Table[0..15]	Vertex Array *Control register*	0xCD00	Bitfield

Bits	Name	Read	Write	Reset	Description
0...12	Tag	✓	✓	x	Tags to be assigned to data in order of output
13..31	Reserved	✓	✗	0	

Notes:

ArrayVertexLoadMask[0,1]

Name	Type	Offset	Format
ArrayVer exLoadMask[0,1]	Vertex Array *Control register*	0xCDC8	Integer

Bits	Name	Read	Write	Reset	Description
0...15	Data	✓	✓	x	Internal stste only read for context switch
16..31	Reserved	✓	✗	0	

Notes:

ArrayVertexStore[0..63]

Name	Type	Offset	Format
ArrayVer exStore[0..63]	Vertex Array *Control register*	0xCF00	Bitfield

Bits	Name	Read	Write	Reset	Description
0...31	Data	✓	✓	x	Internal stste only read for context switch
	Reserved	✓	✗	0	

Notes: **ArrayVertexStore** and **IndexCacheState** hold internal state parms. Which should not be modified or used except to save and restore as part of a context switch.

ArrayVertexStoreValid

Name	Type	Offset	Format
ArrayVer exStoreValid	Vertex Array *Control register*	0xCE80	Bitfield

Bits	Name	Read	Write	Reset	Description
0...15	Data	✓	✕	x	Internal stste only read for context switch
16..31	Reserved	✓	✕	0	

Notes: **ArrayVertexStore** and **IndexCacheState** hold internal state parms. which should not be modified or used except to save and restore as part of a context switch.

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	✔	✔	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.

BackAlpha

Name	Type	Offset	Format
BackAlph a	Material *Control register* Broadcast	0xA930	Float

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

Notes: Alpha value of the back material in floating point format. Normally this is in the range 0.0..1.0. In OpenGL this is usually the back material diffuse alpha.

BackAmbientColorBlue

Name	Type	Offset	Format
BackAmbientColorBlue	Material *Control register* Broadcast	0xA910	Float

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

Notes: Ambient blue color in floating point format for the back material.

BackAmbientColorGreen

Name	Type	Offset	Format
BackAmbientColorGreen	Material *Control register* Broadcast	0xA908	Float

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

Notes: Ambient green color in floating point format for the back material.

BackAmbientColorRed

Name	Type	Offset	Format
BackAmbientColorRed	Material *Control register* Broadcast	0xA900	Float

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

Notes: Ambient red color in floating point format for the back material.

BackDiffuseColorBlue

Name	Type	Offset	Format
BackDiffuseColorBlue	Material *Control register* Broadcast	0xA928	Float

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

Notes: Diffuse blue color for back material in floating point format

BackDiffuseColorGreen

Name	Type	Offset	Format
BackDiff seColorGreen	Material *Control register* Broadcast	0xA920	Float

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

Notes: Diffuse green color for back material in floating point format

BackDiffuseColorRed

Name	Type	Offset	Format
BackDiff seColorRed	Material Ops *Control register* Broadcast	0xA918	Float

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

Notes: Diffuse red color for back material in floating point format

BackEmissiveColorBlue

Name	Type	Offset	Format
BackEmi siveColorBlue	Material *Control register* Broadcast	0xA960	Float

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

Notes: Emissive blue color for back material in floating point format

BackEmissiveColorGreen

Name	Type	Offset	Format
BackEmi siveColorGreen	Material *Control register* Broadcast	0xA958	Float

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

Notes: Emissive green color for back material in floating point format

BackEmissiveColorRed

Name	Type	Offset	Format
BackEmissiveColorRed	Material *Control register* Broadcast	0xA950	Float

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

Notes: Emissive red color for back material in floating point 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.

BackSpecularAlpha

Name	Type	Offset	Format
BackSpecularAlpha	Material *Control register* Broadcast	0xA970	Float

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

Notes: Specular alpha value in floating point format

BackSpecularBlue

Name	Type	Offset	Format
BackSpecularBlue	Material *Control register* Broadcast	0xA948	Float

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

Notes: Specular blue color in floating point format

BackSpecularExponent

Name	Type	Offset	Format
BackSpecularExponent	Lighting *Control register* Broadcast	0xA968	Fixed

Bits	Name	Read	Write	Reset	Description
0...31	Exponent	✓	✓	x	7.4 fixed point number

Notes: This is the back material specular exponent held in unsigned 7.4 fixed point format. This is held in the Lighting Unit even though it is a Material property.

BackSpecularGreen

Name	Type	Offset	Format
BackSpecularGreen	Material; *Control register*	0xA940	Float

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

Notes: Broadcast. Specular green color in floating point format

BackSpecularRed

Name	Type	Offset	Format
BackSpecularRed	Material *Control register* Broadcast	0xA938	Float

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

Notes: Specular red color in floating point format

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 R5'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.

BasePageOfWorkingSetHost

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

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	Vertex Machine Command	0x9590	Bitfield

Bits	Name	Read	Write	Reset	Description
0	AreaStipple Enable	✓	✓	x	Overridden when incompatible with *PrimitiveType*.
1	LineStipple Enable	✓	✓	x	Overridden when incompatible with *PrimitiveType*.
2	ResetLine Stipple	✓	✓	x	*ResetLineStipple* bit is automatically generated so any value provided with the **Begin** is ignored
3	FastFill Enable	✓	✓	x	Ignored, forced to reset
4, 5	Unused	✓	✓	x	
6, 7	Primitive Type	✓	✓	x	Overridden if incompatible with *Type* and Polymode settings.
8	Antialias Enable	✓	✓	x	Qualifies the *AntialiasEnable* bit for each Delta Unit primitive. If both are set the primitive is Antialiased.
9	Antialiasing Quality	✓	✓	x	Ignored - held in the individual primitive mode registers, e.g. LineMode.
10	UsePoint Table	✓	✓	x	Ignored, generated locally when antialiasing pointrs.
11	SyncOnBit Mask	✓	✗	x	Ignored, forced to reset
12	SyncOn Host Data	✓	✓	x	Ignored, forced to reset
13	Texture Enable	✓	✓	x	The TextureEnable bit disables the calculation of S, T and Q from the current texture coordinates and the specular and texture diffuse lighting parameters. It does not disable the texture transformation or texture generation operations in the Geometry and Lighting units to conform with how OpenGL responds.
14	FogEnable	✓	✓	x	Passed through, also enables fog calculations in the Geometry and Lighting units.
15	Coverage Enable	✓	✓	x	Ignored on input, set on output if primitive is antialiased.
16	SubPixel Correction Enable	✓	✓	x	Passed through, but also directs Delta unit to modify rasterizer setup accordingly.
17	Reserved	✓	✓	x	
18	Span Operation	✓	✓	x	Passed through, no effect.
19...27	Unused	✓	✓	x	
28..31	Type	✓	✓	x	0 = null 1 = points 2 = lines 3 = line loop 4 = line strip 5 = triangles 6 = triangle strip 7 = triangle fan 8 = quads 9 = quad strip 10 = polygons

Notes: The main method in OpenGL for issuing primitives is the Begin/End paradigm and the hardware supports this exactly with no additional control by the host. The usual OpenGL commands can be inserted between the Begin and End tags and that is all there is to it. There is no need to manage the vertex store as there was for GLINT Delta.

The Begin tag specifies the primitive type in the upper 4 bits of the data field. The lower bits hold the same fields use by the **Draw*** or **Render** commands in GLINT Delta and GLINT respectively and allow fine control of the texture mapping, fog, etc..

Note: RasterPos should not be sent within a Begin/End sequence (also specified by Ope

The initial values for *vertexCount*, *vertex*, *resetLineStipple*, *objectID* and the various edge flags is passed in as part of the Begin register (in earlier chipsets they were simply reset if necessary). The Vertex Array unit snoops the **Begin** command before rasterization starts to determine the type of primitive being drawn.

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	**Rasterizer 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.

BNx
Bny
BNz

Name	Type	Offset	Format
BNx	Command	0xC910	Bitfield
BNy	Command	0xC908	Bitfield
BNz	Command	0xC900	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0..31	X word	✓	✓	x	BNx component of unpacked blend normal
32..63	Y word	✓	✓	x	BNy component of unpacked blend normal
64..95	Z word	✓	✓	x	BNz component of unpacked blend normal

Notes: This register holds the blend normal as received from the Command Unit. Its tag is changed to what the remaining units are expecting. The normal contains 3 data words for the x,y,z components.

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.

BoundingBoxMaxX

Name	Type	Offset	Format
BoundingBoxMaxX	Matrix	0xC7E8	Int
	Control register		
	Broadcast		

Bits	Name	Read	Write	Reset	Description
0..31	MaxX	✓	✓	x	X value for Bounding Box test

Notes: Holds the max X value for use in the BoundingBox test.

BoundingBoxMaxY

Name	Type	Offset	Format
BoundingBoxMaxY	Matrix	0xC7F0	Int
	Control register		
	Broadcast		

Bits	Name	Read	Write	Reset	Description
0..31	MaxY	✓	✓	x	Y value for Bounding Box test

Notes: Holds the max Y value for use in the BoundingBox test.

BoundingBoxMaxZ

Name	Type	Offset	Format
BoundingBoxMaxZ	Matrix *Control register* Broadcast	0x84F8	Bitfield

Bits	Name	Read	Write	Reset	Description
0..31	MaxY	✓	✓	x	Z value for Bounding Box test

Notes: Holds the max Z value for use in the BoundingBox test.

BoundingBoxMinX

Name	Type	Offset	Format
BoundingBoxMinX	Matrix *Control register* Broadcast	0x84F8	Bitfield

Bits	Name	Read	Write	Reset	Description
0..31	MinX	✓	✓	x	X value for Bounding Box test

Notes: Holds the min X value for use in the BoundingBox test.

BoundingBoxMinY

Name	Type	Offset	Format
BoundingBoxMinY	Matrix *Control register* Broadcast	0x84F8	Bitfield

Bits	Name	Read	Write	Reset	Description
0..31	MinY	✓	✓	x	Y value for Bounding Box test

Notes: Holds the min X value for use in the BoundingBox test.

BoundingBoxMinZ

Name	Type	Offset	Format
BoundingBoxMinZ	Matrix *Control register* Broadcast	0x84F8	Bitfield

Bits	Name	Read	Write	Reset	Description
0..31	MinZ	✓	✓	x	Z value for Bounding Box test

Notes: Holds the min Z value for use in the BoundingBox test.

BoundingBoxTest

Name	Type	Offset	Format
BoundingBoxTest	Matrix *Command register*	0xCA28	Int

Bits	Name	Read	Write	Reset	Description
0..31	In view count	✗	✓	x	Data field contains the number of "in view" fields.

Notes: The Bounding Box test provides an early clip test to discard geometry instructions in a DMA display list which are not visible in the current view. The bounding box test is carried out by writing the following commands and data into a DMA buffer:

- Write the minimum and maximum coordinates of the bounding box into the **BoundingBoxMin**[X, Y, Z] and **BoundingBoxMax**[X, Y, Z] messages.
- Write the 'out of view' display list address and count are written to **DMAFailAddr** and **DMAFailCount** respectively.
- The 'in view' display list address is written to **DMAAddr**.
- Initiate the bounding box test with the **BoundingBoxTest** command (data field holds the 'in view' or pass count).
- Some time later use the **DMAConditionalJump** command to test the result of the bounding box test initiated earlier.

BoundingVertexX

Name	Type	Offset	Format
BoundingVertexX	Matrix *Control register*	0xC898	Int

Bits	Name	Read	Write	Reset	Description
0..31	VertexX	✓	✓	x	Vertex X value

Notes: Holds the X value for use in defining a vertex for the Bounding Volume test.

BoundingVertexY

Name	Type	Offset	Format
BoundingVertexY	Matrix *Control register*	0xC8A0	Int

Bits	Name	Read	Write	Reset	Description
0..31	VertexY	✓	✓	x	Vertex Y value

Notes: Holds the Y value for use in defining a vertex for the Bounding Volume test.

BoundingVertexZ

Name	Type	Offset	Format
BoundingVertexZ	Matrix *Control register*	0xC8A8	Int

Bits	Name	Read	Write	Reset	Description
0..31	VertexZ	✓	✓	x	Vertex Z value

Notes: Holds the Z value for use in defining a vertex for the Bounding Volume test.

BoundingVolumeTest

Name	Type	Offset	Format
BoundingVolumeTest	Matrix *Control register*	0xCA30	Bitfield

Bits	Name	Read	Write	Reset	Description
0..31	In view count	✓	✓	x	

Notes: Before the bounding volume is defined the **StartBoundingVolume** command is issued. The bounding volume (normally a convex hull) is defined a vertex at a time using the **BoundingVertexX**, **BoundingVertexY** and **BoundingVertexZ** registers. The **BoundingVertexZ** register must be updated last as it triggers the bounding vertex to be added to the volume under test.
The bounding volume is terminated using the **EndBoundingVolume** command and this causes the command unit to prompt for the results via the private - command. The bounding volume test is enabled by *BoundingVolumeEnable* bit in the **MatrixMode** register. If the test is disabled then it always returns the bounding box being 'in view'.

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.

BVw

Name	Type	Offset	Format
BVw	Matrix *Control register*	0xC920	Bitfield

Bits	Name	Read	Write	Reset	Description
0..31	Wvalue	✓	✓	x	W component of blended vertex

Notes: The W blended vertex component supplied to build wide vertex data - see **BVx4**

BVx2

Name	Type	Offset	Format
BVx2	Command *Control register*	0xC930	Wide Trigger

Bits	Name	Read	Write	Reset	Description
0..31	X word	✓	✓	x	Bounding vertex X coordinate
32..64	Y word	✓	✓	x	Bounding vertex Y coordinate

Notes: The **BVy**, **BVz** and **BVw** registers hold the floating point value of the y, z and w components of the bounding vertex used to build up the bounding volume for display list viewing frustum culling. The x component is supplied in one of the **BVx2, BVx3** or **BVx4** registers. The numerical postfix defines the which of the **Bvy**, **BVz** and **BVw** registers should be used or have their default values used instead. **BVx2** generates the (x, y, 0.0, 1.0) vertex, **BVx3** generates the (x, y, z, 1.0) vertex and **BVx4** generates the (z, y, z, w) vertex. The **BVx**? register must be loaded last.

BVx3

Name	Type	Offset	Format
BVx3	Command *Control register*	0xC938	

Bits	Name	Read	Write	Reset	Description
0..31	X	✓	✓	x	Bounding vertex X coordinate
32..63	Y	✓	✓	x	Bounding vertex Y coordinate
64..95	Z	✓	✓	x	Bounding vertex Z coordinate

Notes: The **BVy**, **BVz** and **BVw** registers hold the floating point value of the y, z and w components of the bounding vertex used to build up the bounding volume for display list viewing frustum culling. The x component is supplied in one of the **BVx2, BVx3** or **BVx4** registers. The numerical postfix defines the which of the **Bvy**, **BVz** and **BVw** registers should be used or have their default values used instead, e.g. **BVx3** generates the (x, y, z, 1.0) vertex and **BVx4** generates the (z, y, z, w) vertex. The **BVx**? register must be loaded last.

BVx4

Name	Type	Offset	Format
BVx4	Command *Control register*	0xC938	

Bits	Name	Read	Write	Reset	Description
0..31	X	✗	✓	x	Bounding vertex X coordinate
32..63	Y	✗	✓	x	Bounding vertex Y coordinate
64..95	Z	✗	✓	x	Bounding vertex Z coordinate
96..127	w	✗	✓	x	Bounding vertex W coordinate

Notes: The **BVy**, **BVz** and **BVw** registers hold the floating point value of the y, z and w components of the bounding vertex used to build up the bounding volume for display list viewing frustum culling. The x component is supplied in one of the **BVx2, BVx3** or **BVx4** registers. The numerical postfix defines the which of the **Bvy**, **BVz** and **BVw** registers should be used or have their default values used instead, e.g. **BVx2** generates the (x, y, 0.0, 1.0) vertex, **BVx4** generates the (z, y, z, w) vertex. The **BVx**? register must be loaded last.

BVy

Name	Type	Offset	Format
BVy	Command *Control register*	0xC928	Float

Bits	Name	Read	Write	Reset	Description
0..31	Yvalue	✓	✓	x	Y component of blended vertex

Notes: The Y blended vertex component supplied to build wide vertex data - see **BVx4**

BVz

Name	Type	Offset	Format
BVz	Matrix *Control register*	0xC920	Float

Bits	Name	Read	Write	Reset	Description
0..31	Zvalue	✓	✓	x	Z component of blended vertex

Notes: The Z blended vertex component supplied to build wide vertex data - see **BVx4**

Ca

Name	Type	Offset	Format
Ca	Command *Control register*	0x9818	Float

Bits	Name	Read	Write	Reset	Description
0..31	Alpha	✗	✓	x	Alpha

Notes: The alpha color vertex component supplied to build wide vertex data - see **CR4**

Cb

Name	Type	Offset	Format
Cb	Command *Control register*	0x9820	Float

Bits	Name	Read	Write	Reset	Description
0..31	Blue	✗	✓	x	Blue

Notes: The Blue color vertex component supplied to build wide vertex data - see **CR4**

Cg

Name	Type	Offset	Format
Cg	PipeManager *Control register*	0x9828	

Bits	Name	Read	Write	Reset	Description
0..31	Green	✗	✓	x	Green

Notes: The green color vertex component supplied to build wide vertex data - see **CR4**

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*	0x87F0	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
1	Enable	✔	✔	x	This bit, when set, causes the current color to be generated.
2	Shading	✔	✔	x	Selects the shading mode. The two options are: 0 = Flat – the color is taken from the Constant Color register. 1 = Gouraud – the color is taken from the DDAs.
3…31	Unused	0	0	x	

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

ColorMaterialMode
ColorMaterialModeAnd
ColorMaterialModeOr

Name	Type	Offset	Format
ColorMaterialMode	Material	0x9528	Bitfield
ColorMaterialModeAnd	Material	0xAAC0	Bitfield
ColorMaterialModeOr	Material	0xAAC8	Bitfield
	Control registers		
	Broadcast		

Bits	Name	Read[9]	Write	Reset	Description
0	ColourMaterial Enable	✔	✔	x	When set causes a colour register to update the material parameter(s) for the given face(s).
1,2	Face	✔	✔	x	This field selects which face(s) any material changes should be made to by the Colour register. The values are: 0 front material 1 back material 2 front and back material

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

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

3..5	Parameter	✔	✔	x	This field selects which material parameter(s) should updated by a Colour Register. The values are: 0 Emissive 1 Ambient 2 Diffuse 3 Specular 4 Ambient and diffuse
6,7	FrontEmissiveSource	✔	✔	x	This field selects where the front emissive material comes from. The options are: 0 Front material emissive 1 Current colour 2 Current diffuse 3 Current specular
8…9	FrontAmbientSource	✔	✔	x	As above, but option 0 is Front ambient material.
10…11	FrontDiffuseSource	✔	✔	x	As above, but option 0 is Front diffuse material.
12…13	FrontSpecularSource	✔	✔	x	As above, but option 0 is Front specular material.
14…15	FrontAlphaSource	✔	✔	x	As above, but option 0 is Front Alpha material.
16…17	BackEmissive Source	✔	✔	x	As above, but option 0 is Back Emissive material
18…19	BackAmbient Source	✔	✔	x	As above, but option 0 is Back Ambient material
20…21	BackDiffuseSource	✔	✔	x	As above, but option 0 is Back diffuse material.
22…23	BackSpecularSource	✔	✔	x	As above, but option 0 is Back specular material.
24…25	BackAlphaSource	✔	✔	x	As above, but option 0 is Back alpha material.
26…27	FrontSpecularAlphaSource	✔	✔	x	As above, but option 0 is Front specular alpha material.
28…29	BackSpecularAlphaSource	✔	✔	x	As above, but option 0 is Back specular alpha material.

Notes: Material parameters can be edited directly or by using the Colour registers[10] (not to be confused with the **Colour** register in GLINT). The **ColourMaterialMode** specifies which parameters should be updated by a **Colour** register. Two sets of material parameters are held - one set for front faces and one set for back faces.

[10]The user actually uses the C* registers or a packed colour register and the Vertex Machinary Unit issues the Colour register.

CommandDMAControl
CommandDMAControlAnd
CommandDMAControlOr

Name	Type	Offset	Format
Comman lDMAControl	Command	0xCA00	Bitfield
Comman lDMAControlAnd	Command	0xCA08	Bitfield
Comman lDMAControlOr	Command	0xCA10	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	Protocol	✓	✓	x	0 = PCI 1 = AGP
1	Alignment	✓	✓	x	0 = Off 1 = Align to 64-byte boundary where possible
2, 3	ByteSwap	✓	✓	x	0 = ABCD (no swap) 1 = BADC 2 = CDAB 3 = DCBA
4..6	BurstSize	✓	✓	x	Log2 of size of DMA burst in Dwords.
7..31	Reserved	✓	✕	0	

Notes: No readback from And/Or logical variants.

CommandFilter

Name	Type	Offset	Format
Comman lFilter	Command	0xCE78	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0..7	Data	✓	✓	x	Bit field of chips to have command streams ignored
8..31	Reserved	0	0	x	

Notes:

CommandID

Name	Type	Offset	Format
Comman lID	Vertex Array *Control register*	0xA990	Bitfield

Bits	Name	Read	Write	Reset	Description
0…31	Data	0	0	x	User defined field

Notes: The **CommandID** can be used to mark a point in the command stream; it is a programmable register that can be read at any time without syncing the chip. It is typically used to report which buffers have been completed and can be reused. It takes effect after all DMA commands issued before it have completed.

CommandInterrupt

Name	Type	Offset	Format
Comman lInterrupt	Rectangle DMA *Control register*	0xA990	Bitfield

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

Notes: **CommandInterrupt** generates an interrupt. If bit 0 is Set (=1) the interrupt is held until all previously started rectangle DMAs have completed. Command processing is also suspended while waiting.

Config2D

Name	Type	Offset	Format
Config2D	Global *Control register*	0xB618	Bitfield

Bits	Name	Read	Write	Reset	Description
0	Opaque Span	✓	✓	x	In *RasterizerMode*, *Scissor Mode*, *LogicalOpMode*, *FBWriteMode*, *TextureReadMode*.
1	Reserved	✗	✗	x	Reserved
2	UserScissorEnable	✓	✓	x	*ScissorMode*
3	FBDestReadEnable	✓	✓	x	In *FBDestReadMode* bit 3 = (ReadEnable)
4	AlphaBlendEnable	✓	✓	x	In *AlphaBlendColorMode* and *AlphaBlendAlphaMode*: bit 4 = AlphaBlendEnable (Enable)
5	DitherEnable	✓	✓	x	In *DitherMode*: bit 5 = DitherEnable (Enable)
6	ForgroundLogicalOpEnable	✓	✓	x	In *LogicalOpMode*: bit 6 = ForgroundLogicalOpEnable (Enable)
7…10	ForgroundLogicalOp	✓	✓	x	In *LogicalOpMode*: Bits 7-10 = ForgroundLogicalOp (LogicOp)
11	BackgroundLogicalOpEnable	✓	✓	x	In *LogicalOpMode*: Bit 11 = BackgroundLogicalOpEnable (Background En.)
12…15	BackgroundLogicalOp	✓	✓	x	In *LogicalOpMode*: Bits 12-15 = BackgroundLogicalOp
16	UseConstantSource	✓	✓	x	In *LogicalOpMode*: bit 16 = UseConstantSource
17	FBWriteEnable	✓	✓	x	In *FBWriteMode*: bit 17 = FBWriteEnable (WriteEnable)
18	Blocking	✓	✓	x	In *FBSourceReadMode* bit 18 = Blocking
19	ExternalSourceData	✓	✓	x	In *FBSourceReadMode* bit 19 = ExternalSourceData
20	LUTMode Enable	✓	✓	x	In *LUTMode*: bit 20 = Enable

Notes: This register updates the mode registers in multiple units as shown. The name in brackets is the field name in the corresponding mode register, if different to the field name for the *Config2D* command. Also note that bit 0 affects several mode registers.

Constant Color

Name	Type	Offset	Format
ConstantColor	Delta *Control register*	0x87E8	Bitfield

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

Notes: This register holds the constant color in packed format and provides the constant color when the DDAs are not in use. This register has been largely superceded by the **ConstantColorDDA** register.

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: 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.

ContextData

Name	Type	Offset	Format
ContextData	Global *Control register*	0x8DD0	Variable

Bits		Read	Write	Reset	Description
0…31	ContextData	✓	✕	x	Undefined, returned by ContextDump command = (number of data words) -1

Notes: Following a **ContextDump**, the data written out takes this tag. Context data can be restored the same way, i.e. using **ContextRestore** with the same context mask will read data from the **ContextData** register. The actual data format is largely undocumented and should not be edited.

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	116
1	Geometry	✗	✔	x	Delta unit state	574
2	Matrices	✗	✔	x		130
3	Material	✗	✔	x		29
4	Lights0_7	✗	✔	x		200
5	Lights8_15	✗	✔	x		200
6	RasterPos	✗	✔	x		24
7	CurrentState	✗	✔	x		61
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)	27
10	Select	✗	✔	x		67
11	RasterizerState	✗	✔	x	General setup of the rasterization units	226
12	DDA	✗	✔	x	DDA Values	69
13	Ownership	✗	✔	x	Stripe ownership state	3
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	Multi-texture	✗	✔	x		0
18	PipeControl	✗	✔	x		320
19	MatrixStack	✗	✔	x		0
20	Vertex Array	✗	✔	x		169
21…31	Reserved	✗	✔	x		0

Notes: This command forces R5 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
- If a context dump is done with only the CurrentState bit or the RasterPos bit set, then the context buffer provides the following information which can be queried by (e.g.) OpenGL

CurrentState=1	
Offset	***Data***
0	Edge Flag
1	Normal X component
2	Normal Y component
3	Normal Z component
4	Texture S component
5	Texture T component
6	Texture R component
7	Texture Q component
8	Current Red
9	Current Green
10	Current Blue
11	Current Alpha

RasterPos=1	
Offset	***Data***
0	Window Coordinate X
1	Window Coordinate Y
2	Window Coordinate Z
3	Eye Coordinate, Z
4	Clip coordinate, W
5	Texture S
6	Texture T
7	TextureR
8	Texture Q
9	Fog
10	InView (0=no, 1=yes)
11	XIncrement (user reg)
12	YIncrement (user reg)
13	XOffset (user reg)
14	YOffset (user reg)
15	Color Red component
16	Color Green component
17	Color Blue component
18	Color Alpha component

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	116
1	Geometry	✗	✔	x	Delta unit state	574
2	Matrices	✗	✔	x		130
3	Material	✗	✔	x		29
4	Lights0_7	✗	✔	x		200
5	Lights8_15	✗	✔	x		200
6	RasterPos	✗	✔	x		24
7	CurrentState	✗	✔	x		61
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)	27
10	Select	✗	✔	x		67
11	RasterizerState	✗	✔	x	General setup of the rasterization units	226
12	DDA	✗	✔	x	DDA Values	69
13	Ownership	✗	✔	x	Stripe ownership state	3
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	Multi-texture	✗	✔	x		0
18	PipeControl	✗	✔	x		320
19	MatrixStack	✗	✔	x		0
20	Vertex Array	✗	✔	x		169
21…31	Reserved	✗	✔	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.

Cr3

Name	Type	Offset	Format
Cr3	Command *Control register*	0x9830	Integer

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

Notes: Provides the red vertex color component and triggers packing of a 3 word r,g,b wide with this tag. See also **Cb, Cg**

Cr4

Name	Type	Offset	Format
Cr4	Command *Control register*	0x9838	

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

Notes: Provides the red vertex color component and triggers packing of a 4 word r,g,b,a wide with this tag. See also Cb, Cg, Ca.
The nominal range for color components is 0.0 to 1.0, but this is not strictly enforced. The **CR3** or **CR4** register must be written last as it writes the wide with values defined to that point into the pipeline. Components should be written together, not interleaved with writes to other registers. The color may be used as the vertex color if lighting is disabled, or to change a material property if ColorMaterial is disabled. See **ColorMaterialMode**, **LightingMode**, **MaterialMode**.

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 gradient 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.

DataSync

Name	Type	Offset	Format
DataSync	RectsngleDMA *Control register*	0xCEC0	Integer

Bits	Name	Read	Write	Reset	Description
0..31	Count	✓	✓	x	Count of 32 bit words to transfer

Notes:

DataSyncTarget

Name	Type	Offset	Format
DataSync Target	RectsngleDMA *Control register*	0xCEC8	Bitfield

Bits	Name	Read	Write	Reset	Description
0..12	Tag	✓	✓	x	Tag to use with Sync Data
13..31	Reserved	✕	✕	0	

Notes:

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 gradient 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 dominant Y gradient 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 [11]	Write	Reset	Description
0, 1	TargetChip	✓	✓	x	The bitfield is retained for backwards compatibility but in R5 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 R5. 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	SubPixel Correction Enable	✓	✓	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 R5. 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
18	Reserved	✓	✓	x	Reserved

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

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	This field instructs the Delta unit on the method to be used in the colorDDA to do flat shading: 0 = Use **ConstantColor** register value 1 = Use DDA with zero gradients Rasterization performance is not affected, but **ConstantColor** is faster to set up. .
23...31	Reserved	0	0	x	Reserved

Notes:
- Constantcolor Flat Shading is faster to set up than DDA flat shading but there is otherwaise no performance difference. This field would normally be the inverse of the *FlatShading* field in **GeometryMode**
- 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.

DeltaModeOverride0
DeltaModeOverride1

Name	Type	Offset	Format
DeltaMo eOverride0	Delta	0xCED8	Integer
DeltaMo eOverride0	Delta	0xCEE0	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…31	User defined data	✓	✓	x	

Notes: Enabled in **TextureFormatControl**, allows elimination of duplicated tasks when using emulated dual texturing.

DeltaSwitchMode
DeltaSwitchModeAnd
DeltaSwitchModeOr

Name	Type	Offset	Format
DeltaSwitchMode	Delta	0xAFE8	Bitfield
DeltaSwitchModeAnd	Delta	0xAFF0	Bitfield Logic Mask
DeltaSwitchModeOr	Delta	0xAFF8	Bitfield Logic Mask
	Control registers		

Bits	Name	Read [12]	Write	Reset	Description
0	ForceNoDraw	✓	✓	x	When set, modifies DeltaMode* messages as they are forwarded to force the NoDraw bit to be set
1	PmaskFilter	✓	✓	x	When set, enables tag filtering based on the pmask field of incoming tags
2	StripeSelect	✓	✓	x	When cleared, only tags with bit 0 of Pmask set are valid. When set, only tags with bit 1 of Pmask set are valid. PmaskFilter bit must also be set to enable this behaviour.
3	FilterAll	✓	✓	x	When set, filters out all outgoing messages except SetDeltaPort.
4	FilterPrimitives	✓	✓	X	When set, prevents any Render commands from being forwarded
5, 6	Clamp ILLEnable	✓	✓	X	Enables clamping of the Inverse Line Length: 0 = no clamping 1 = clamp only when stippling is enabled 2 = Clamp whether stippling is enabled or not.
7..14	MaxILL Exponent	✓	✓	X	Sets the exponent to clamp the InverseLineLength values against when ClampILLEnable is set.

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

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 [13]	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.

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

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…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 gradient 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 [14]	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.

[14] 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.

DKsGdx

	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: 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: 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: The X gradient value for the Red 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: he 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	Command Control Register	0xA980	Integer

Bits	Name	Read	Write	Reset	Description
0…1	Reserved	0	0	X	
2…31	Address	✔	✔	X	Address

Notes: A secondary DMA is initiated by the **DMAAddr** and **DMACount** commands. The base address of the DMA buffer is loaded by **DMAAddr**, which must be aligned to a 32-bit boundary. The data field of the **DMACount** command sets the number of 32 bit words to read and acts as the trigger for the DMA.

- 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**.

DMABoundingBoxJump

Name	Type	Offset	Format
DMABounding BoxJump	Command *Command*	0xCA38	Integer

Bits	Name	Read	Write	Reset	Description
0…29	Count	✗	✔	x	Number of DMA words to transfer
30…31	Reserved	0	0	x	

Notes: The bounding box test is used to quickly reject DMA buffers that contain no geomtery that is visible from the current view position. When the **DMABoundingBoxJump** command is executed it uses the results of the bounding box test to conditionally execute the DMA defined by **DMAAddr** (as usual) and the count in the data field of **DMABoundingBoxJump** (instead of **DMACount**). If the test passes, the DMA is performed, but if the test fails a different DMA is invoked using data from registers **DMAFailAddr** and **DMAFailCount**

DMAContinue

Name	Type	Offset	Format
DMACon tinue	Command *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: The DMAContinue command is used as a lightweight DMA that concatenates with the DMA currently in progress. It holds a count as the data field that is used to extend the current DMA, so small DMAs may be combined into long bursts for better bus efficiency. To avoid the overhead of polling the input FIFO space every time a DMAContinue command is sent, the continue is absorbed immediately and added to the current count, so taking no space in the input FIFO. To enable double buffering of commands two separate address/count pairs are maintained so that a new address may be loaded and continues applied to it while the current DMA is executing, and still take no space in the FIFO.

Note: Write Combining

If something is mapped in as write-combined the CPU can empty its write-combine buffer in any order. R5 supports a 64 byte buffer (16 dwords) so the lower 4 bits of the tag represent the position within the write combine buffer; G3 uses them to put the data back to the correct order. When used in this mode, the CPU writes data to incrementing addresses, but the data may be sent to R5 jumbled up.

The normal mode of operation is that a start address for the DMA buffer is loaded, then DMAContinue commands are sent until the buffer has been completed, when a new address is sent. Using packets to load the DMAContinue command is inefficient because it takes two words (header plus continue). To make this more efficient any data written to the address space marked as write-combine capable is interpreted as a DMAContinue command, and does not need a header.

Actually write-combining the FIFO is a function of the CPU. If it is not enabled then the one word continue command still operates. If write-combining is enabled all writes must be to consecutive incrementing addresses and the first combined write following a non-combined write must be to an address on a 32 byte boundary..

DMACount

Name	Type	Offset	Format
DMACount	Command *Control register*	0xA988	Integer

Bits	Name	Read	Write	Reset	Description
0…29	Count	✔	✔	x	Number of DMA words to transfer
30…31	Reserved	0	0	x	

Notes: At chip reset the *MasterEnable* bit in the **CFGCommand** register must be set to allow DMA to operate. Then, for the simplest form of DMA, the host software prepares a host buffer containing register address tag descriptions and data values. The host writes the base address of this buffer to the **DMAAddr** register and the count of the number of words to transfer to the **DMACount** register. Writing to the **DMACount** register starts the DMA transfer and the host is then free to perform other work. New DMAs can be nested to 2 levels. If 2 DMAs are active then subsequent **DMACount** tags are ignored.

*Note: **DMACount** should not be confused with the PCI register of the same name.*

DMACurrentAddress[0..15]

Name	Type	Offset	Format
DMACurrentAddress[0..15]	Command *Control register*	0xD500	Integer

Bits	Name	Read	Write	Reset	Description
0..1	Reserved	✓	✕	0	
2..31	Address	✓	✓	x	Current position in DMA buffer

Notes: The **DMACurrentAddress** register returns the position in a DMA that has been reached. It guarantees that the memory address it points to has been read and can be modified by the CPU. There is one register for each command unit. Some registers are specific to a unit and are identified by a number added to the end of the register name. For example, the **DMACurrentAddress** register is independently addressable in each command unit, so there is **DMACurrentAddress0**, **DMACurrentAddress1**, etc.

DMAFailAddr

Name	Type	Offset	Format
DMAFail Addr	Command Control Register	0xCA18	Integer

Bits	Name	Read	Write	Reset	Description
0…1	Reserved	0		X	
2…31	Address	✔	✔	X	Address

Notes: If the **DMABoundingBox** test fails a different DMA is invoked using data from registers **DMAFailAddr** and **DMAFailCount**. If **DMAFailCount** is set to zero then a failure does nothing and execution continues as though the DMA buffer was not present. The purpose of the fail DMA is to make it possible to set the state of the graphics system to what it would have been if the test had passed (note that the DMA buffer can contain commands that change state as well geometry data).

DMAFailCount

Name	Type	Offset	Format
DMAFail Count	Command *Control register*	0xCA20	Integer

Bits	Name	Read	Write	Reset	Description
0…29	Count	✔	✔	x	Number of DMA words to transfer
30…31	Reserved	0	0	x	

Notes: See **DMAFailAddr**

DMAFeedback

Name	Type	Offset	Format
DMAFeedback	RectangleDMA *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. It starts a new output DMA if the output DMA is idle. Otherwise it blocks until the output DMA controller becomes available. This suspends all subsequent commands and register loads.

- The feedback DMA transfer is set up by using the **DMAOutputAddress** register (to define the memory buffer address) and the **DMAFeedback** command.
- To allow the output DMA to collect data from the Host Out FIFO, bits 14 and 15 of the **FilterMode** register must also be set.
- 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. The Host Out FIFO is read until the **EndOfFeedback** tag is encountered or buffer meory is exceeded.
- The actual number of words written to memory is loaded into the **FeedbackSelectCount** PCI register

DMAOutputAddress

Name	Type	Offset	Format
DMAOutputAddress	Rectangle DMA *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. See **DMAFeedback.**

DMAOutputCount

Name	Type	Offset	Format
DMAOu putCount	Rectangle DMA *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 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 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, DMAFeedback**

DMAProfile

Name	Type	Offset	Format
DMAPro ile	Vertex Array *Control Register*	0x?	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Count	✔	✔	x	Count of time spent waiting on full FIFO

Notes:

DMARectangleReadControl
DMARectangleReadControlAnd
DMARectangleReadControlOr

Name	Type	Offset	Format
DMARectangleReadControl	RectangleDMA	0xCA48	Integer
DMARectangleReadControl	RectangleDMA	0xCA50	Integer
DMARectangleReadControl	RectangleDMA	0xCA58	Integer

Control Register

Bits	Name	Read	Write	Reset	Description
0	Protocol	✔	✔	x	0 = PCI, 1 = AGP
1	Alignment	✔	✔	x	0 = off, 1 = align to 64 byte boundary where possible
2-3	ByteSwap	✔	✔	x	0 = ABCD (no swap) 1 = BADC 2 = CDAB 3 = DCBA
4-6	BurstSize	✔	✔	x	Log2 of size of DMA burst in DWords
7-31	Reserved	✔	✕	x	

Notes:

DMARectangleRead

Name	Type	Offset	Format
DMARectangleRead	RectangleDMA *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 lines. 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..31	Reserved	0	0	x	

Notes:
1. The Rectangle DMA mechanism allows image data to be transferred from host memory to R5. 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 R5 it can be in packed or unpacked format..
3. The minimum number of PCI reads are used to align and pack the image data.
4. R5 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; DMARectangleReadControl; DMARectangleReadLinePitch; DMARectangle ReadTarget**.

DMARectangleReadAddress

Name	Type	Offset	Format
DMARectangleReadAddress	RectangleDMA *Control Register*	0xA9B0	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Address	✔	✔	x	32 bit pixel aligned byte 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 R5. 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**; **DMARectangleReadControl**.

DMARectangleReadControl
DMARectangleReadControlAnd
DMARectangleReadControlOr

Name	Type	Offset	Format
DMARectangleReadControl	RectangleDMA	0xCA48	Bitfield
DMARectangleReadControlAnd	RectangleDMA	0xCA50	Bitfield
DMARectangleReadControlOr	RectangleDMA	0xCA58	Bitfield
	Control Register		

Bits	Name	Read	Write	Reset	Description
0	Protocol	✓	✓	x	0 = PCI 1 = AGP
1	Alignment	✓	✓	x	0 = Off 1 = Align to 64 byte boundary where possible
2, 3	ByteSwap	✓	✓	x	0 = ABCD (no swap) 1 = BADC 2 = CDAB 3 = DCBA
4..6	BurstSize	✓	✓	x	Log2 of size of DMA burst in Dwords.
7..31	Reserved	0	0	x	

Notes:

DMARectangleReadLinePitch

Name	Type	Offset	Format
DMARectangleReadLinePitch	RectangleDMA	0xA9B8	Integer
	Control Register		

Bits	Name	Read	Write	Reset	Description
0…31	Line Pitch	✔	✔	x	Pixel aligned byte count

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 R5. 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	RectangleDMA *Command*	0xA9C0	Bitfield

Bits	Name	Read	Write	Reset	Description
0-12 Tag	Tag	✔	✔	x	Tag to use with DMA data.
13-31	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 R5. 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	RectangleDMA *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..31	Reserved	0	0	x	

Notes:
1. The Rectangle DMA mechanism allows image data to be transferred from R5 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. R5 is set up to rasterize the source area for the pixel data (depth, stencil, color, etc.) enabled in the Render command. This is done before the Rectangular DMA is started.
5. This register must be loaded by itself and not as part of any increment, hold or indexed group.
6. See also: **DMARectangleReadAddress**; **DMARectangleReadLinePitch**; **DMARectangleReadTarget**

DMARectangleWriteAddress

Name	Type	Offset	Format
DMARectangleWriteAddress	RectangleDMA	0xA9D0	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…31	Address	✔	✔	x	32 bit pixel aligned byte 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 R5 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*

DMARectangleWriteControl
DMARectangleWriteControlAnd
DMARectangleWriteControlOr

Name	Type	Offset	Format
DMARectangleWriteControl	RectangleDMA	0xCA68	Bitfield
DMARectangleWriteControlAnd	RectangleDMA	0xCA70	Bitfield
DMARectangleWriteControlOr	RectangleDMA	0xCA78	Bitfield
	Control Register		

Bits	Name	Read	Write	Reset	Description
0	Reserved	✓	✗	0	
1	Alignment	✓	✓	x	0 = Off 1 = Align to 64 byte boundary where possible
2, 3	ByteSwap	✓	✓	x	0 = ABCD (no swap) 1 = BADC 2 = CDAB 3 = DCBA
4..31	Reserved	0	0	x	

Notes:

DMARectangleWriteLinePitch

Name	Type	Offset	Format
DMARectangleWriteLinePitch	Input	0xA9D8	Integer
	Control Register		

Bits	Name	Read	Write	Reset	Description
0…31	Line Pitch	✔	✔	x	LinePitch in pixels

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 R5 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** register to set up the word address. Each subsequent **DownloadData** sends 32 bits of register 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. For a more efficient approach to bitmaps see DMARectangle

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** register. Each **DownloadData** command sends 32 bits of register 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 gradient 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	HostIn *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, draws a line from vertex 1 to vertex 0. The vertices are loaded separately.
This tag is not used by the geometry engine. It is a HostIn instruction which affects the rasterizer only and is retained for backwards compatibility.

DrawLine10

Name	Type	Offset	Format
DrawLine01	HostIn *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.
- This tag is not used by the geometry engine. It is a HostIn instruction which affects the rasterizer only and is retained for backwards compatibility

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. This tag is not used by the geometry engine and affects only the rasterizer - it is retained for backwards compatibility.

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 gradient 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 gradient 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. The register was formerly 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. The register was formerly 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 gradient 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 dominant Y gradient 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
dXDom	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
dXSub	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	✓	✗	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.

EdgeFlag

Name	Type	Offset	Format
EdgeFlag	Vertex machine *Control register*	0x95A0	boolean

Bits	Name	Read	Write	Reset	Description
0	Flag	✓	✓	x	0=false 1=true
1..31	Unused	✓	✓	x	

Notes: Sets the current edge flag to the value in the least significant bit. Edge flags are single bit booleans, but are also held in 32 bit words. Although this is inefficient they are rarely loaded through a vertex array.

End

Name	Type	Offset	Format
End	Delta unit *Command register*	0x9598	

Bits	Name	Read	Write	Reset	Description
0…31	End	✗	✗	x	

Notes: Terminates a Begin/End pair defining a primitive to render. Tag only, no associated data. **End** simply consumes the register since it is not used subsequently. See **Begin**.

EndBoundingVolume

Name	Type	Offset	Format
EndBoundingVolume	Matrix *Control register*	0xC888	bitfield

Bits	Name	Read	Write	Reset	Description
0..3	channel	✗	✓	x	Selects return channel
4..31	Unused	✗	✓	x	

Notes: The bounding volume is terminated using the **EndBoundingVolume** command which causes the Command unit to prompt for the results. This register is used by the Command Unit to return the results of the bounding volume test. The bottom 4 bits specify which of 16 return channels the results should be posted in.

EndOfFeedback

Name	Type	Offset	Format
EndOfFeedback	RectangleDMA *Command*	0x8FF8	unused

Bits	Name	Read	Write	Reset	Description
0	EndofFeedback	✗	✔	x	Command tag

Notes: DMA transfers to or from the 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 **ContextDump** 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.
- *Caution*
  When using Feedback DMA mode, if Run Length Encoding is NOT enabled the chip can lock up because the **EndOfFeedback** tag cannot be detected.

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	✓	✓	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 [15]	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.

[15] 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 [16]	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 R5.
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 R5.
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.
16…17	Layout2	✔	✔	x	Selects the layout of the pixel data in memory for buffer 2.

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

18…19	Layout3	✔	✔	x	Selects the layout of the pixel data in memory for buffer 3.
20 21 22 23	Origin0 Origin1 Origin2 Origin3	✔	✔	x	These fields selects where the window origin is for buffer 0…3 respectively. The options are: 0 = Top Left. 1 = Bottom Left
24	Blocking	✔	✔	x	This bit, when set, causes destination span reads to block to prevent reads and writes from overlapping (in time). Each span is read in full and then written. This is less efficient than streaming (bit is clear), but allows overlapping blits (spans overlap) without corruption. Note this does not need to be set if the destination read and write buffers are the same.
25	Reserved	0	0	x	
26	UseRead Enables	✔	✔	x	When this bits is set the enables in the FBDestReadEnables register are used to determine if a destination read is required. The ReadEnable bit must also be set and the corresponding buffer bits as well for a read to occur.
27	Alpha Filtering	✔	✔	x	This bit, when set, compares the fragment's alpha value and if it is equal to the AlphaReference value (held in the FBReadEnables register) then no read is done. This is done to save memory bandwidth when the alpha blend mode is such that with the given alpha value the destination color doesn't contribute to the fragment's color.

Notes: The destination address calculation(s) are controlled by the FBDestReadMode register and the address is a function of X, Y, FBDestReadBufferAddr, FBDestReadBufferOffset, FBDestReadBufferWidth and PixelSize parameters. The Addr, Offset and Width are specified independently for each of the four possible write buffers.

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

FBHardwareWriteMask

Name	Type	Offset	Format
FBHardwareWriteMask	Framebuffer *Control register*	0x8AC0	

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

Notes: This register holds the write mask used for all writes. When a bit is set the corresponding bit in each framebuffer word is set (enabled for writing). The masking is actually done in the memory devices so has zero impact on performance and doesn't require any reads.

- The hardware write mask applies only where 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 register*	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 [17]	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 R5.
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 R5.
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

[17] 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 [18]	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 R5.

[18] 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 R5.
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.

FeedbackAlpha

Name	Type	Offset	Format
Feedback Alpha	Material *Control register*	0x8FC0	Integer

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

Notes: This register holds the feedback Alpha colour value.

FeedbackBlue

Name	Type	Offset	Format
Feedback Blue	Material *Control register*	0x8FB8	Integer

Bits	Name	Read	Write	Reset	Description
0...31	color	✕	✓	X	32 bit integer

Notes: This register holds the feedback Blue colour value.

FeedbackGreen

Name	Type	Offset	Format
Feedback Green	Material *Control register*	0x8FB0	Integer

Bits	Name	Read	Write	Reset	Description
0...31	color	✕	✓	x	32 bit integer

Notes: This register holds the feedback Green colour value.

FeedbackRed

Name	Type	Offset	Format
Feedback Red	Material *Control register*	0x8FA8	Integer

Bits	Name	Read	Write	Reset	Description
0...31	color	✕	✓	x	32 bit integer

Notes: This register holds the feedback Red colour value.

FeedbackQ

Name	Type	Offset	Format
Feedback Q	Texture/Fog *Control register*	0x8FE0	Integer

Bits	Name	Read	Write	Reset	Description
0...31	texture	✕	✓	x	32 bit integer

Notes: This register holds the feedback Q texture value.

FeedbackR

Name	Type	Offset	Format
Feedback R	Texture/Fog *Control register*	0x8FD8	Integer

Bits	Name	Read	Write	Reset	Description
0...31	Texture R	✕	✓	x	

Notes: This register holds the feedback R texture value

FeedbackS

Name	Type	Offset	Format
Feedback S	Texture/Fog *Control register*	0x8FC8	Integer

Bits	Name	Read	Write	Reset	Description
0...31	Texture S	✕	✓	x	

Notes: This register holds the feedback S texture value.

FeedbackT

Name	Type	Offset	Format
Feedback T	Texture/Fog *Control register*	0x8FD0	Integer

Bits	Name	Read	Write	Reset	Description
0...31	Texture T	✕	✓	x	

Notes: This register holds the feedback T texture value.

FeedbackToken

Name	Type	Offset	Format
FeedbackToken	Geometry *Control register*	0x8F80	Integer

Bits	Name	Read	Write	Reset	Description
0...31	Primitive	✗	✓	x	

Notes: This message defines the primitive type the following feedback data describes. The number of vertices the data represents depends on the primitive type and this is encoded as follows:

0x44e02000	Point
0x44e04000	Line
0x44e06000	Triangle
0x44e08000	Bitmap
0x44e0a000	DrawPixel
0x44e0c000	CopyPixel
0x44e0e000	LineReset

FeedbackW

Name	Type	Offset	Format
FeedbackW	Geometry *Control register*	0x8FA0	Integer

Bits	Name	Read	Write	Reset	Description
0...31	Texture W	✗	✓	x	

Notes: This register holds the feedback W coordinate value

FeedbackX

Name	Type	Offset	Format
FeedbackX	Geometry *Control register*	0x8F88	Integer

Bits	Name	Read	Write	Reset	Description
0...31	X	✗	✓	x	32 bit integer

Notes: This register holds the feedback X coordinate value

FeedbackY

Name	Type	Offset	Format
Feedback Y	Geometry *Control register*	0x8F90	Integer

Bits	Name	Read	Write	Reset	Description
0...31	Y	✕	✓	x	32 bit integer

Notes: This register holds the feedback Y coordinate value

FeedbackZ

Name	Type	Offset	Format
Feedback Z	Geometry *Control register*	0x8F98	Integer

Bits	Name	Read	Write	Reset	Description
0...31	Z	✕	✓	x	32 bit integer

Notes: This register holds the feedback Z coordinate value.

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	2D Setup	0x8338	Bitfield
FillConfig2D1	2D Setup *Control register*	0x8360	Bitfield

Bits	Name	Read	Write	Reset	Description
0	Opaque Span	✕	✓	x	In **RasterizerMode, Scissor Mode, LogicalOpMode, FBWriteMode, TextureReadMode**.
1	~~MultiRXBlit~~	✕	✓	x	**RasterizerMode, ScissorMode** - reserved on R5
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	2DSetup *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	2D Setup *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	2D Setup *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	2D Setup *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 [19]	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.

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

15	RemainderData	✔	✔	x	When set allows any data not covered by the categories in this table to be written into the output FIFO.
16…17	ByteSwap	✔	✔	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 (i.e. in Gamma). 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. When this is detected in the DMA controller the DMA can be terminated.

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 register 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** register. 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 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.

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:

FNx

Name	Type	Offset	Format
FNx	Matrix *Control register*	0x84F8	Bitfield

Bits	Name	Read	Write	Reset	Description
0..31	X word	✓	✓	x	FNx component of unpacked face normal
32..63	Y word	✓	✓	x	FNy component of unpacked face normal
64..95	Z word	✓	✓	x	FNz component of unpacked face normal

Notes: This register holds the face normal as received from the Command Unit. Its tag is changed to what the remaining units are expecting. The normal contains 3 data words for the x,y,z components.

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..

FogDensity

Name	Type	Offset	Format
FogDens ty	Texture/Fog *Control register* Broadcast	0x9BB0	Float

Bits	Name	Read	Write	Reset	Description
0…31	Density	✔	✔	x	Float

Notes: Fog density as a floating point number..

FogEnd

Name	Type	Offset	Format
FogEnd	Texture/Fog *Control register* Broadcast	0x9BC0	Float

Bits	Name	Read	Write	Reset	Description
0…31	fogend	✔	✔	x	Float

Notes: Fog end as a floating point number.

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 [20]	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.

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

FogScale

Name	Type	Offset	Format
FogScale	Texture/Fog *Control register* Broadcast	0x9BC0	Float

Bits	Name	Read	Write	Reset	Description
0…31	fogscale	✔	✔	x	Float

Notes: Fog scale as a floating point number. This is 1/(fog end - fog start) value.

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 *Control registers*	0xB280…B2F8	Bitfield

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

FogVertexMode
FogVertexModeAnd
FogVertexModeOr

Name	Type	Offset	Format
FogVerte :Mode	Texture/Fog	0x9538	Bitfield
FogVerte :ModeAnd	Texture/Fog	0xAB50	Bitfield Logic Mask
FogVerte :ModeOr	Texture/Fog	0xAB58	Bitfield Logic Mask

Control registers

Broadcast

Bits	Name	Read [21]	Write	Reset	Description
0	Enable	✔	✔	x	When set causes the fog value associated with the vertex to be calculated and sent to the Delta Unit. It is qualified by the FogEnable bit in the Begin register. The actual method of calculating the fog value is given by the Function field.
2, 3	Function	✔	✔	x	This field selects the function used to calculate the fog value as a function of the parameter selected by the Sourc field. The options are: 0: Linear 1: Exponential 2: ExponentialSquared 3: None (i.e. just use input value)
4,5	Source	✔	✔	x	This field selects the souce of the parameter to use in the fog calculations. The options are: 0: EyeZ 1: Range (i.e. magnitude of the eye vertex) 2: User 3: Projected Z value
13…31	Reserved	0	0	x	Reserved

Notes:
- Provides the mode control information for the fog calculation.
- The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

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

ForegroundColor

Name	Type	Offset	Format
ForegroundColor	LogicOps *Control register*	0xB0C0	Integer

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

Notes: With **BackgroundColor**, holds the foreground and background color values. The color format is in the raw framebuffer format and 8 or 16 bit pixels are automatically replicated to fill the 32 bits of register.

FrontAlpha

Name	Type	Offset	Format
FrontAlpha	Material *Control register* Boradcast	0xA8B0	Float

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

Notes: Alpha value of the primitive in floating point format..

FrontAmbientColorBlue

Name	Type	Offset	Format
FrontAmbientColorBlue	Material *Control register* Boradcast	0xA890	Float

Bits	Name	Read	Write	Reset	Description
0…31	Blue	✓	✓	x	blue

Notes: Blue value of the primitive in floating point format..

FrontAmbientColorGreen

Name	Type	Offset	Format
FrontAmbientColorGreen	Material *Control register* Boradcast	0xA888	Float

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

Notes: Green value of the primitive in floating point format..

FrontAmbientColorRed

Name	Type	Offset	Format
FrontAmbientColorRed	Material *Control register* Boradcast	0xA880	Float

Bits	Name	Read	Write	Reset	Description
0…31	Red	✓	✓	x	red

Notes: Alpha value of the primitive in floating point format..

FrontDiffuseColorBlue

Name	Type	Offset	Format
FrontDiffuseColorBlue	Material *Control register* Boradcast	0xA8A8	Float

Bits	Name	Read	Write	Reset	Description
0…31	Blue	✓	✓	x	blue

Notes: Blue value of the primitive in floating point format..

FrontDiffuseColorGreen

Name	Type	Offset	Format
FrontDif useColorGreen	Material *Control register* Boradcast	0xA8A0	Float

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

Notes: Green value of the primitive in floating point format..

FrontDiffuseColorRed

Name	Type	Offset	Format
FrontDif useColorRed	Material *Control register* Boradcast	0xA898	Float

Bits	Name	Read	Write	Reset	Description
0…31	Red	✓	✓	x	red

Notes: Red value of the primitive in floating point format..

FrontSpecularAlpha

Name	Type	Offset	Format
FrontSpe ularAlpha	Material *Control register* Boradcast	0xA8F0	Float

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

Notes: Alpha value of the primitive in floating point format..

FrontSpecularColorBlue

Name	Type	Offset	Format
FrontSpecularColorBlue	Material *Control register* Boradcast	0xA8C8	Float

Bits	Name	Read	Write	Reset	Description
0…31	Blue	✓	✓	x	blue

Notes: Blue value of the primitive in floating point format..

FrontSpecularColorGreen

Name	Type	Offset	Format
FrontSpecularColorGreen	Material *Control register* Boradcast	0xA8C0	Float

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

Notes: Green value of the primitive in floating point format..

FrontSpecularColorRed

Name	Type	Offset	Format
FrontSpecularColorRed	Material *Control register* Boradcast	0xA8B8	Float

Bits	Name	Read	Write	Reset	Description
0…31	Red	✓	✓	x	red

Notes: Red value of the primitive in floating point format..

FrontSpecularExponent

Name	Type	Offset	Format
FrontSpecularExponent	Lighting *Control register* Broadcast	0xA8E8	Fixed

Bits	Name	Read	Write	Reset	Description
0...31	Exponent	✓	✓	x	7.4 fixed point number

Notes: This is the front material specular exponent held in unsigned 7.4 fixed point format. This is held in the Lighting Unit even though it is a Material property.

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.

GammaFilter

Name	Type	Offset	Format
GammaFilter	Matrix *Control register* Broadcast	0x9578	

Bits	Name	Read	Write	Reset	Description
0…30	Mask	✓	✓	x	
31	Enable	✓	✓	x	1 = enable

Notes: Holds a mask used to filter all downsteam data forwarding. If the bit assigned for this Gamma (defined external to this unit via pins or a config eeprom) is clear then no tags are forwarded. Bit 31 must be set for the filtering to have an effect.

GeometryMode
GeometryModeAnd
GeometryModeOr

Name	Type	Offset	Format
Geometry Mode	Geometry	0x9510	Bitfield
Geometry Mode And	Geometry	0xAA90	Bitfield Logic Mask
Geometry Mode Or	Geometry	0xAA98	Bitfield Logic Mask

Control registers

Broadcast

Bits	Name	Read [22]	Write	Reset	Description
0	SendInvW	✔	✔	x	This bit, when set, causes the coordinate information sent to the Delta Unit to include 1/w value. The message used for this is DeltaXYZW[0…3], otherwise DeltaCoord[0…3] is used.
1	Reserved	0	0	x	
2	UserInvW	✔	✔	x	This bit, when set, takes the invW value from the w field on the input vertex rather calculating the reciprocal of w itself. A typical use of this is in D3D when post transformed and clipped vertices are provided as (x, y, z, 1/w) in screen coordinates.
3	Orthographic Projection	✔	✔	x	This bit, when set, does not do the perspective division as part of the view port mapping. This is also useful in D3D where the w value provided with post transformed and clipped vertices is in fact 1/w so its use in the perspective division would give the wrong result as the vertices have already been projected.
4, 5	FrontPoly Mode	✔	✔	x	This field selects the how a triangle, quad or polygon should be drawn when its orientation is facing forwards. The options are: 0: Point 1: Line 2: Fill
6, 7	BackPoly Mode	✔	✔	x	This field selects the how a triangle or quad or polygon should be drawn when its orientation is facing backwards. The options are: 0: Point 1: Line 2: Fill
8	FrontFace Direction	✔	✔	x	This field selects which direction is the 'front' facing direction. The direction is important as it is used to determine if a triangle, etc. should be culled (if enabled), the material to use during lighting, and the PolyMode to use. 0: Clockwise 1: Counter Clockwise

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

9	PolygonCull	✔	✔	x	This field, when set, enabled polygon culling based on the front face direction. It is ignored for points, lines and rectangles.
10, 11	PolygonCull Face	✔	✔	x	This field determines which direction of face should be culled (if enabled). It has the following values: 0: Front 1: Back 2: Front and Back
12	ClipShortLines	✔	✔	x	Clipping is an expensive operation and in some cases (for short lines) it is much faster to draw them and rely on the window and/or screen clipping during rasterisation. When this bit is 0 lines below the length given in the LineClipThreshold message are not clipped. This does not apply if the line crosses the near, far or user clipping planes.
13	ClipSmall Triangles	✔	✔	x	Clipping is an expensive operation and in some cases (for small triangles) it is much faster to draw then and rely on the window and/or screen clipping during rasterisation. When this bit is 0 triangles below the 'area' given in the TriangleClipThreshold message are not clipped. This does not apply if the triangle crosses the near, far or user clipping planes.
14,15	RenderMode	✔	✔	x	The RenderMode field controls the action when processing any primitive. The options are: 0: Render 1: Select 2: Feedback
16…18	FeedbackType	✔	✔	x	This field only has any effect if the RenderMode is Feedback. In this case it determines the parameters to be returned for every primitive. The options are: 0: X, Y 2D 1: X, Y, Z 3D 2: X, Y, Z, R, G, B, A 3Dcolour 3: X, Y, Z, R, G, B, A, S, T, R, Q 3DColourTexture 4: X, Y, Z, W, R, G, B, A, S, T, R, Q 4DColourTexture
19	CullUsingFace Normal	✔	✔	x	This field, if set will cull using the supplied face normal (in the FaceNormal message). The face normal does not have to be of unit length. If no face normal is supplied then the normal method of backface culling is used. This is only used in the Cull Unit.

20	DisableFustum Culling	✔	✔	x	This bit, when set, disabled fustum culling. This prevents screen based vertex data to be sent with a W of 1.0 (maybe added automatically) from being clipped out. This is useful for D3D as the transformation process can be disabled, unlike OpenGL where it is always enabled.
21	FlatShading	✔	✔	x	When selects flat shading to be used, otherwise Gouraud shading will be used.
22… 27	UserClipMask	✔	✔	x	There is one bit per user defined clipping plane. Clipping against a plane is enabled when the corresponding bit is set. The clipping plane is defined in eye space by UserClipn{X | Y | Z | W}. Bit 0 (i.e. bit 22 in register) corresponds to UserClip0.
28	Polygon OffsetPoint	✔	✔	x	This field, if set, causes the polgyon offset to be calculated and applied to the points of a polyogn when PolyMode is set to Point.
29	Polygon OffsetLine	✔	✔	x	This field, if set, causes the polgyon offset to be calculated and applied to the lines of a polyogn when PolyMode is set to Line.
30	Polygon OffsetFill	✔	✔	x	This field, if set, causes the polgyon offset to be calculated and applied to the triangles of a polyogn when PolyMode is set to Fill.
31	InvertFace NormalCull Direction	✔	✔	x	This field, if set, causes the supplied Face Normal to be inverted before it is used for backface culling.

Notes:

GeomRectangle

Name	Type	Offset	Format
GeomRectangle	Geometry *Control register*	0x96A0	Bitfield

Bits	Name	Read	Write	Reset	Description
0, 1	Type	✕	✓	x	These two bits define the type of rectangle to be inserted into the feedback buffer. They have not effect when not in feedback mode. The options are: 0 Bitmap 1 DrawPixel 2 CopyPixel 3 Don't insert into the feedback buffer.
2	OffsetEnable	✕	✓	x	When this bit is set the x and y offset values held in RasterPosXOffset and RasterPosYOffset respectively displace to the raster position window coordinates when the rectangle is rendered. This does not update the raster position state.
3	SelectEnable	✕	✓	x	When this bit is set the rectangle takes part in the selection process.
4..31	Reserved	✕	0	x	

Notes:
- The GeomRectangle command is used to render the rectangle, based on the position, colour, etc. established with the write to the RPx register. If the raster position is not in view then all GeomRectangle command are ignored (with the exception of the increment action - see later). If the raster position is in view then the operation is controlled by the data fields above.
- After every rectangle is submitted using the GeomRectangle command (in any RenderMode) the window coordinate x and y components is updated by the amount held in the RasterPosXIncrement and RasterPosYIncrement registers respectively.

GIDMode
GIDModeAnd
GIDModeOr

Name	Type	Offset	Format
GIDMode	Localbuffer	0xB538	Bitfield
GIDMode And	Localbuffer	0xB5B0	Bitfield Logic Mask
GIDMode Or	Localbuffer	0xB5B8	Bitfield Logic Mask
	Control registers		

Bits	Name	Read [23]	Write	Reset	Description
0	Fragment Enable	✔	✔	x	When set, does GID testing on fragments. If the test fails then the fragment is discarded
1	Span Enable	✔	✔	x	When set, GID tests/modifies the span pixel mask pixel-by-pixel. Pixels which fail the test are disabled.
2…5	Compare Value	✔	✔	x	Holds the 4 bit GID comparison reference value. Set unused bits (where the GID width in the local buffer format is less than 4 bits) to zero.
6…7	Compare Mode	✔	✔	x	This field holds the comparison modes available for use during GID testing. The options are: 0 = Always pass 1 = Never pass (i.e. always fail) 2 = Pass when local buffer gid == CompareValue 3 = Pass when local buffer gid != CompareValue
8…9	Replace Mode	✔	✔	x	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	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: Enabling GraphicID span testing causes this unit to read the GIDs corresponding to the pixels in the span, do the test and modify the pixel mask appropriately. The test is done 4 pixels at a time and the resulting span can be then used as normal, or used to clear the local buffer, thereby clearing local buffer pixels at 4 times the single fragment rate.
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.

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

GlyphData

Name	Type	Offset	Format
GlyphData	2DSetup *Control register*	0xB660	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Packed data	✗	✔	x	Glyph data byte stream

Notes: A byte stream of glyph data (packed four to a word) can be downloaded and automatically chopped up and padded to the necessary width for the texture units to use as a bitmap. For example a gyph with a width between 17 and 24 pixels will be sent down as a stream of bytes and each triplet of bytes will be padded with zero and sent to be written into memory. If the input words have their bytes labelled:

First word: DCBA (A is the least significant byte)
Second word: HGFE

Then the output words send on to the rasterizer are:

First word: 0CBA
Second word: 0FED

GlyphPosition

Name	Type	Offset	Format
GlyphPosition	2DSetup *Control register*	0xB608	Integer

Bits	Name	Read	Write	Reset	Description
0…15	X offset	✓	✓	x	2's complement X coordinate
16…31	Y offset	✓	✓	x	2's complement Y coordinate

Notes: This register defines the glyph origin for use by the **Render2DGlyph** command. This register is updated by the **Render2DGlyph** command and the updated values will be read back or context dumped.

GStart

Name	Type	Offset	Format
Gstart	Color *Control register*	0x8798	Fixed point

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.

IncrementObjectID

Name	Type	Offset	Format
Incremen ObjectID	PipeManager *Command*	0x95B0	Integer

Bits	Name	Read	Write	Reset	Description
0...15	ObjectID	✗	✓	x	Current ObjectID number in 16 bits
16..31	Unused	✓	✓	x	

Notes: The command **IncrementObjectID** increments the stored *ObjectID* value and then informs the rasteriser about the new *ObjectID* value. The *ObjectID* rolls over to zero when its range is exceeded.

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:

InitNames

Name	Type	Offset	Format
InitNames	Geometry *Command*	0x95D8	

Bits	Name	Read	Write	Reset	Description
0..31	Unused	✕	✕	x	

Notes: This command initialises the Name stack - only the tag is used.

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.

InvertCurrentMatrix

Name	Type	Offset	Format
InvertCurrentMatrix	Matrix *Command*	0xC6C8	Tag

Bits	Name	Read	Write	Reset	Description
0…31	Unused	0	0	x	Unused

Notes: This command causes the **CurrentMatrix** to be inverted and the result stored in **IMat** and written to the **NormalMatrix**. The upper 3x3 matrix of the transposed **IMat** is copied into the **NormalMatrix**.

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: 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 *Control registers*	0xB598	Bitfield Logic Mask

Bits	Name	Read [24]	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

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

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
LBDestReadModeOr	Localbuffer	0xB588	Bitfield Logic Mask
	Control registers		

Bits	Name	Read [25]	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.
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..
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.

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

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
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 [26]	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	

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

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..
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..
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 [27]	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..
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..
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.

[27] 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.

LightAmbientIntensityBlue

Name	Type	Offset	Format
LightAmbientIntensityBlue	Lighting *Control register* Broadcast	0xD598	Float

Bits	Name	Read	Write	Reset	Description
0...31	Blue	✓	✓	x	32 bit float

Notes: Ambient blue intensity in floating point format..

LightAmbientIntensityGreen

Name	Type	Offset	Format
LightAmbientIntensityGreen	Lighting *Control register* Broadcast	0xD590	Float

Bits	Name	Read	Write	Reset	Description
0...31	Green	✓	✓	x	32 bit float

Notes: Ambient green intensity in floating point format..

LightAmbientIntensityRed

Name	Type	Offset	Format
LightAmbientIntensityRed	Lighting *Control register* Broadcast	0xD588	Float

Bits	Name	Read	Write	Reset	Description
0...31	Red	✓	✓	x	32 bit float

Notes: Ambient red intensity in floating point format.. The light stores are written by first loading up the **LightNumber** register with the light to update and then the desired parameters are written.

LightConstantAttenuation

Name	Type	Offset	Format
LightConstantAttenuation	Lighting *Control register* Broadcast	0xD618	Float

Bits	Name	Read	Write	Reset	Description
0...31	Blue	✓	✓	x	32 bit float

Notes:

- Set when the light's position has a non zero W (this indicates the position holds a position and not a direction) and the sum of the attenuation factors is not unity. ConstantAttenuation is defined on a per-light basis.
- There is enough local storage to hold the parameters for 16 lights per pipe on chip. The light stores are written by first loading up the **LightNumber** register with the light to update and then the desired parameters are written. To read a light the desired pipe and light number are written to the PCI **LightControl** register and then the desired parameters read. The programmer need not concern themselves that the lights are distributed across multiple units
- For details of attenuation factors refer to the *OpenGL Programming Guide*, "Creating Light Sources".

LightCosSpotlightCutoffAngleInner

Name	Type	Offset	Format
LightCos potlight CutoffAn gleInner	Lighting *Control register* Broadcast	0xD630	Float

Bits	Name	Read	Write	Reset	Description
0...31	Red	✓	✓	x	32 bit float

Notes: Cosine of the spotlight cutoff inner angle. Its range is 0.0 to 1.0 inclusive. D3D spotlights only..

LightCosSpotlightCutoffAngleOuter

Name	Type	Offset	Format
LightCos potlight CutoffAn gleOuter	Lighting *Control register* Broadcast	0xD610	Float

Bits	Name	Read	Write	Reset	Description
0...31	Red	✓	✓	x	32 bit float

Notes: Cosine of the spotlight cutoff inner angle. Its range is 0.0 to 1.0 inclusive. D3D spotlights only..

LightCosSpotlightInvSlope

Name	Type	Offset	Format
LightCos potlightInvSlope	Lighting *Control register* Broadcast	0xD638	Float

Bits	Name	Read	Write	Reset	Description
0...31	Red	✓	✓	x	32 bit float

Notes: Cosine of the spotlight cutoff inner angle. Its range is 0.0 to 1.0 inclusive. D3D spotlights only..

LightDiffuseIntensityBlue

Name	Type	Offset	Format
LightDiffuseIntensityBlue	Lighting *Control register* Broadcast	0xD5B0	Float

Bits	Name	Read	Write	Reset	Description
0...31	Blue	✓	✓	x	32 bit float

Notes: Diffuse blue intensity in floating point format..

LightDiffuseIntensityGreen

Name	Type	Offset	Format
LightDiffuseIntensityGreen	Lighting *Control register* Broadcast	0xD5A8	Float

Bits	Name	Read	Write	Reset	Description
0...31	Green	✓	✓	x	32 bit float

Notes: Diffuse green intensity in floating point format..

LightDiffuseIntensityRed

Name	Type	Offset	Format
LightDiffuseIntensityRed	Lighting *Control register* Broadcast	0xD5A0	Float

Bits	Name	Read	Write	Reset	Description
0...31	Red	✓	✓	x	32 bit float

Notes: Diffuse red intensity in floating point format..

LightingMode
LightingModeAnd
LightingModeOr

Name	Type	Offset	Format
LightingMode	Delta	0x9520	Bitfield
LightingModeAnd	Delta	0xAAB0	Bitfield Logic Mask
LinodeOr	Delta	0xAAB8	Bitfield Logic Mask

Control registers

Broadcast

Bits	Name	Read [28]	Write	Reset	Description
0	Enable	✔	✔	x	When set causes the vertex to be lit using the lighting equations, otherwise the current colour is assigned.
1,2	TwoSided Lighting	✔	✔	x	The three options are: 0 Use the front side materials. 1 Use the back side materials and invert the normal before it is used in the lighting calculations. 2 Use the orientation of the face to select between front or back materials and lighting. The orientation is determined by the geometry processing. Fields in the GeometryMode register control the association between a front face and the vertex ordering. When Select is used most vertices will only be lit on one side, however vertices shared between a front facing and back facing polygon will be lit twice. Changing a material or light parameter during a primitive may cause the outstanding vertices to be lit on both sides if Select is used.
3	LocalViewer	✔	✔	x	When set causes the viewer to be at (0, 0, 1) in eye coordinates, otherwise the viewer is at (0, 0, ∞). When the viewer is at infinity some of the lighting equations can be simplified and so run faster, however the position of the specular highlights are not as correct.
4	reserved	✘	✔	x	
5	RangeTest	✔	✔	x	When set forces the lighting calculation for the current light to be aborted when the light's range exceeds the per light defined value held in the range parameter. This optimisation allows lights which are becoming too faint to contribute to be terminated early.
6…14	reserved	✘	✔	x	
15	Specular LightingEnable	✔	✘	x	When this bit is set the specular part of the lighting calculations are done, otherwise they are skipped. When this bit is 0 it overrides the specular material monitoring.

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

16	reserved	✕			
17	D3Deye Direction	✔	✔	x	This bit, when set, causes the eye direction to be (0, 0, -1) otherwise the OpenGL convention of (0, 0, 1) is used.
18	Calculate Half Vector	✔	✔	x	This bit, when set, causes the normalised half vector for non spot lights to be calculated whenever the light's position is updated. It leaves the half vector in the light's spot light direction registers. The eye direction as determined by the D3DEyeDirection bit.
19	D3DSpotlight	✔	✔	x	This bit, when set, causes the D3D spot light light fall off calculations to be used instead of the OpenGL calculations.
20	Fast Normalisation	✔	✔	x	This bit, when set, causes the specular angle normalisation in the slow lighting case to be done to less precision to make the operation faster.
21..31	Reserved	✕	✔	x	

Notes: Controls the loading and calculation of lights.
There are 16 lighting units which calculate lights in parallel.

LightLinearAttenuation

Name	**Type**	**Offset**	**Format**
LightLinearAttenuation	Lighting *Control register* Broadcast	0xD620	Float

Bits	**Name**	**Read**	**Write**	**Reset**	**Description**
0...31	Attenuation	✓	✓	x	32 bit float

Notes: Linear attenuation factor. For details of attenuation factors refer to the *OpenGL Programming Guide*, "Creating Light Sources".

LightMode

Name	Type	Offset	Format
LightMode	Lighting *Control register* Broadcast	0xD580	Bitfield

Bits	Name	Read	Write	Reset	Description
0	LightOn	✓	✓	x	When set indicates the light is on and contributes illumination to the scene, otherwise it does not.
1	Spotlight	✓	✓	x	When set indicates the light is a spotlight. If it is not set then the light is not a spot light and the SpotlightDirection is used to hold the normalised half vector between the viewer and the light. OpenGL indicates a light is not a spotlight when the spotlight cut off angle is set to 180.0 degrees.
2	Attenuation	✓	✓	x	When set indicates the light is to be attenuated, otherwise no attenuation is done. OpenGL indicates a light is not to be attenuated when the W component of the light's position is zero.
3	LocalLight	✓	✓	x	When set indicates the light is local and the full lighting equations should be used. This allows a light to be local without it having to be a spotlight or have any attenuation applied to it.

Notes: This field holds the per light mode information..

LightNumber

Name	Type	Offset	Format
LightNumber	Lighting *Control register* Broadcast	0xC780	

Bits	Name	Read	Write	Reset	Description
0..5	Number	✓	✓	x	Light number
6...31	Reserved	✗	✓	x	

Notes: This message holds the light number light number to be updated by the light parameters. The bottom 6 bits hold the light number.

LightPositionW

Name	Type	Offset	Format
LightPosi ionW	Lighting *Control register* Broadcast	0xD5E8	Float

Bits	Name	Read	Write	Reset	Description
0...31	PositionW	✓	✓	x	32 bit float

Notes: W position of the light. When zero, it changes the meaning of Position* values to directions. Holds 1/w for non zero w values.

LightPositionX

Name	Type	Offset	Format
LightPosi ionX	Lighting *Control register* Broadcast	0xD5D0	Float

Bits	Name	Read	Write	Reset	Description
0...31	PositionX	✓	✓	x	32 bit float

Notes: X position of the light if PositionW = 0, otherwise it is the normalised X direction..

LightPositionY

Name	Type	Offset	Format
LightPosi ionX	Lighting *Control register* Broadcast	0xD5D8	Float

Bits	Name	Read	Write	Reset	Description
0...31	PositionY	✓	✓	x	32 bit float

Notes: Y position of the light if PositionW = 0, otherwise it is the normalised Y direction..

LightPositionZ

Name	Type	Offset	Format
LightPosi ionZ	Lighting *Control register* Broadcast	0xD5E0	Float

Bits	Name	Read	Write	Reset	Description
0...31	PositionX	✓	✓	x	32 bit float

Notes: Z position of the light if PositionW = 0, otherwise it is the normalised Z direction..

LightQuadraticAttenuation

Name	Type	Offset	Format
LightQua draticAttenuation	Lighting *Control register* Broadcast	0xD628	Float

Bits	Name	Read	Write	Reset	Description
0...31	Attenuation	✓	✓	x	32 bit float

Notes: Linear attenuation factor. For details of attenuation factors refer to the *OpenGL Programming Guide*, "Creating Light Sources".

LightRange

Name	Type	Offset	Format
LightRan ;e	Lighting *Control register* Broadcast	0xD640	Float

Bits	Name	Read	Write	Reset	Description
0...31	Attenuation	✓	✓	x	32 bit float

Notes: Linear attenuation factor. For details of attenuation factors refer to the *OpenGL Programming Guide*, "Creating Light Sources".

LightSpecularIntensityBlue

Name	Type	Offset	Format
LightSpecularIntensity Blue	Lighting *Control register* Broadcast	0xD5C8	Float

Bits	Name	Read	Write	Reset	Description
0...31	Blue	✓	✓	x	32 bit float

Notes: Specular blue intensity in floating point format..

LightSpecularIntensityGreen

Name	Type	Offset	Format
LightSpecularIntensity Green	Lighting *Control register* Broadcast	0xD5C0	Float

Bits	Name	Read	Write	Reset	Description
0...31	Green	✓	✓	x	32 bit float

Notes: Specular green intensity in floating point format..

LightSpecularIntensityRed

Name	Type	Offset	Format
LightSpecularIntensityRed	Lighting *Control register* Broadcast	0xD5B8	Float

Bits	Name	Read	Write	Reset	Description
0...31	Red	✓	✓	x	32 bit float

Notes: Specular red intensity in floating point format..

LightSpotlightDirectionX

Name	Type	Offset	Format
LightSpo lightDirection X	Lighting *Control register* Broadcast	0xD5F0	Float

Bits	Name	Read	Write	Reset	Description
0...31	X	✓	✓	x	32 bit float

Notes: Normalised X component for the spotlight direction or the normalised X component of the half vector when the light is not a spotlight..

LightSpotlightDirectionY

Name	Type	Offset	Format
LightSpo lightDirection Y	Lighting *Control register* Broadcast	0xD5F8	Float

Bits	Name	Read	Write	Reset	Description
0...31	Y	✓	✓	x	32 bit float

Notes: Normalised Y component for the spotlight direction or the normalised Y component of the half vector when the light is not a spotlight..

LightSpotlightDirectionZ

Name	Type	Offset	Format
LightSpo lightDirectionZ	Lighting *Control register* Broadcast	0xD600	Float

Bits	Name	Read	Write	Reset	Description
0...31	Z	✓	✓	x	32 bit float

Notes: Normalised Z component for the spotlight direction or the normalised Z component of the half vector when the light is not a spotlight..

LightSpotlightExponent

Name	Type	Offset	Format
LightSpo lightExponent	Lighting *Control register* Broadcast	0xD608	Float

Bits	Name	Read	Write	Reset	Description
0...31	Exponent	✓	✓	x	32 bit float

Notes: Spotlight exponent, controls the intensity distribution of light within a cone. This is held as an unsigned 7.4 fixed point number.. For details of attenuation factors refer to the *OpenGL Programming Guide*, "Spotlights".

LineClipLengthThreshold

Name	Type	Offset	Format
LineClipI engthThreshold	Geometry *Control register* Broadcast	0x9BD8	Float

Bits	Name	Read	Write	Reset	Description
0...31	Exponent	✓	✓	x	32 bit float

Notes: The length of a line in pixels, projected onto the x or y axis, for which geometric clipping will ***not*** be done if enabled.

LineExtend

Name	Type	Offset	Format
LineExte d	Cull *Control register* Broadcast	0xC5A0	Fixed

Bits	Name	Read	Write	Reset	Description
0...31	Scanlines	✓	✓	x	8.3 unsigned fixed point.

Notes: This message hold the number of scanlines to grow the y extent by when dealing with lines. It is updated whenever the *LineWidth* value in the Delta Unit is updated.

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 [29]	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** 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** 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** 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** register.
27	AntialiasEnable	✔	✔	x	This field, when set, enables antialiasing of lines. This is qualified by the *AntialiasEnable* field in the **Begin** register.
28	Antialiasing Quality	✔	✔	x	This field defines the quality of antialiased points: 0 4x4 1 8x8
29...31	Reserved	✔	✕	x	

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.

29 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.

LoadName

Name	Type	Offset	Format
LoadNan e	Geometry *Command*	0x95F0	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Name	✕	✔	x	Name

Notes: **LoadName** replaces the entry on the top of the name stack with the value in the data field. See also **InitNames**.

The **GeometryMode** command allows selection of the mode of operation (*RenderMode*), which is usually Render but can also be Feedback or Select. Select mode implements the OpenGL feature with that name and is used for Picking and Selection of primitives. Select mode is broken down into two parts:
• Primitives are clipped and culled but not rendered. If a primitive passed the clipping and culling phases then a hit flag is set and the minimum and maximum Z range grown, if necessary, to include this primitive. Subsequent primitives which also pass the clipping and backface culling may extend the minimum and/or maximum Z values.
• Name stack manipulation. The name stack holds names (as 32 bit integers) the user can push, pop or load to keep track of the model hierarchy. The commands **PushName**, **PopName**, **LoadName** do these actions. **PopName** and **LoadName** update the stack with the 32 bit value in the data field. The name stack is reset with the **InitNames** command. The name stack is 64 entries deep.

If the hit flag is set when a name stack manipulation is done a hit record is written to the host output FIFO. The hit flag is then reset along with the minimum and maximum Z range. The hit record consists of (in order):

- The count of the names on the stack (+ some error flags),
- The minimum Z value as a normalised floating point number,
- The maximum Z value as a normalised floating point number,
- The name stack entries, oldest first (variable number [0…64] of words.

Bits 14 and 15 in the **FilterMode** register must be set to allow the *SelectRecord* tag and data values to be written in to the FIFO. The name stack manipulations commands are ignored when not in Select mode.

For details about the use of the Hit record and how it is parsed for OpenGL purposes refer to the *GLINT R5 Programmer's Guide*, volume I.

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 [30]	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.

[30] 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…31	Unused	0	0	x	

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

LogicalTexturePageTableAddr

Name	Type	Offset	Format
LogicalTexturePageTableAddr	Texture	0xB4D0	Integer
	Control register		

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

Notes: This register holds the base address of the Logical Texture Page Table. The address should be aligned to a 64 bit boundary.

LogicalTexturePageTableLength

Name	Type	Offset	Format
LogicalTexturePageTableLength	Texture	0xB4D8	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0…16	Logical page count	✓	✓	x	17 bit integer value from 0 to 65536

Notes: This register holds the number of logical pages to be managed. Any logical pages past this value are folded to logical page 0. Setting this register to zero effectively disables logical to physical mapping. The legal range of values is 0…65536.

LUT[0…15]

Name	Type	Offset	Format
LUT[0..15]	LUT *Control registers*	0x8E80	Bitfield

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

Notes: These registers allow the lower 16 entries of the LUT to be loaded and read back directly.

LUTAddress

Name	Type	Offset	Format
LUTAddress	Texture *Control register*	0x84D0	Integer

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

Notes: This register holds the physical address of a block of data to load into the LUT from memory. This is given as a byte address, but the bottom 4 bits are ignored so the address is effectively aligned to a 128 bit memory word.

LUTData

Name	Type	Offset	Format
LUTData	Texture *Control register*	0x84C8	Integer

Bits	Name	Read	Write	Reset	Description
0…31	LUT data word	✓	✓	x	32 bit value

Notes: This register holds the 32 bits of data to load into the LUT. The data can be loaded in 'as is', have its red and green components swapped over or converted into a replicated 16 bit format.
LUT readback is done by first reading the **LUTIndex** register. As well as returning the current LUT index it has the additional effect of setting the ReadIndex counter to zero. The ReadIndex counter is only used during readback and is not the same as the LUTIndex used for loading the LUT via the pipeline. 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 register is written. The index is automatically incremented after each load and wraps from 255 to 0.

Note: Readback from the LUTIndex has the side effect of clearing the ReadIndex

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 [31]	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

[31] 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** register with the index in the bits 0…7 and the count in bits 8…13. The **LUTTransfer** register 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.

MaterialMode
MaterialModeAnd
MaterialModeOr

Name	Type	Offset	Format
MaterialMode	Material *Command* Broadcast	0x9530	Bitfield
MaterialModeAnd	Material *Command* Broadcast	0xAB10	Bitfield
MaterialModeOr	Material *Command* Broadcast	0xAA98	Bitfield

Bits	Name	Read	Write	Reset	Description
0	Enable	✔	✔	x	When set causes the vertex to calculated from the material values and light colour otherwise the current colour is assigned.
1	DiffuseTextureEnable	✔	✔	x	When set allows the diffuse texture colour to be calculated and sent to GLINT
2	SpecularTextureEnable	✔	✔	x	When set allows the specular texture colour to be calculated and sent to GLINT
3	Reserved	✔	✔	x	
4	Reserved	✔	✔	x	
5	Premultiply Alpha	✔	✔	x	When set premultiplies the diffuse and ambient colours by the selected alpha value.
6	Reserved	✔	✔	x	
7, 8	TwoSided Lighting	✔	✔	x	The three options are: 0 Use the front side materials. 1 Use the back side materials. 2 Use the orientation of the face to select between front or back materials and lighting.
9	SpecularColour	✔	✔	x	This bit, when set, modifies the equations for colour and specular textures to match the OpenGL 1.2 specification.

10	SendNormal	✔	✔	x	This bit, when set, causes the vertex normals to be send to the Delta Unit to support bump mapping. The normals are also clipped and renormalised along with the colour, etc. parameters when the primitive is clipped.
11	SendColourA	✔	✔	x	This bit, when set, causes the vertex colour (as received and not computed) to be sent to the Delta Unit. This will be used as an additional colour itterator in future rasteriser chips.
12	SendColourB	✔	✔	x	This bit, when set, causes the vertex diffuse parameter (as received and not computed) to be sent to the Delta Unit. This will be used as an additional colour itterator in future rasteriser chips.
13	SendColourC	✔	✔	x	This bit, when set, causes the vertex specular parameter (as received and not computed) to be sent to the Delta Unit. This will be used as an additional colour itterator in future rasteriser chips.
14…15	ColourSource	✔	✔	x	This field selects where the colour message comes from when the Enable field is 0. The options are: 0 Front material diffuse 1 Current colour 2 Current diffuse 3 Current specular
16…17	DiffuseTexture Source	✔	✔	x	This field selects where the DiffuseTexture message comes from when the Enable field is 0. The options are: 0 Front material ambient 1 Current colour 2 Current diffuse 3 Current specular
18…19	SpecularTextur eSource	✔	✔	x	This field selects where the colour message comes from when the Enable field is 0. The options are: 0 Front material specular 1 Current colour 2 Current diffuse 3 Current specular
20…21	AlphaSource	✔	✔	x	This field selects where the colour message comes from when the Enable field is 0. The options are: 0 Front material diffuse 1 Current colour 2 Current diffuse 3 Current specular

22	ColourDisable	✔	✔	x	This bit, when set, prevents the colour message from being sent out. This is only useful when the colour is known be be constant over a range of vertices so can be preloaded in the Delta Unit or rasteriser. This occurs in CDRS and ProCDRS benchmarks where white antialiased lines are the predominant primitive type. Disabling the transmission of the colour information reduces the bandwidth for these tests from 13 words per line to 9 words per line.
23…24	SpecularAlphaSource	✔	✔	x	This field selects where the specular alpha value comes from when the Enable field is 0. The options are: 0 Front material specular alpha 1 Current colour 2 Current diffuse 3 Current specular
25	IncludeSpecularAlpha	✔	✔	x	This bit, when set, causes an alpha value to be included in the specularTexture value. In this case the DeltaColourC* messages are used, otherwise the DeltaTextureKs* messages are used (these are needed for backwards compatibility as current rasterisers only expect 3 component specular textures).
26..31	Reserved	✘	✔	x	

Notes: The Material Unit does vertex coloring, processes normals, sends vertex colors to the Delta unit and loads and edits material parameters. Most of this is handled by the **MaterialMode** register.
The Material Unit is told when to send vertex colour information to the Delta Unit by the **VertexColour** or **ClippedColour** command under the direction of the Geometry Unit.

MatrixMode
MatrixModeAnd
MatrixModeOr

Name	Type	Offset	Format
MatrixMode	Matrix	0x9570	Bitfield
MatrixModeAnd	Matrix	0xC600	Bitfield
MatrixModeOr	Matrix	0xC608	Bitfield
	Command		

Bits	Name	Read	Write	Reset	Description
0..5	Reserved	✔	✔	x	
6	BoundingBox TestEnable	✔	✔	x	This bit, when set, enables the bounding box test to be applied to the bounding box previously loaded into the BoundingBoxMin[X, Y, Z] and BoundingBoxMax[X, Y, Z] registers.
7	ColourOrder	✔	✔	x	This bit selects the PackedFormatByte order for the PackedColour, PackedSpecular and PackedDiffuse messages. The options are: 0 = BGR 1 = RGB
8	Bounding VolumeTest Enable	✔	✔	x	This bit, when set, enables the bounding volume test to be done.
9,10	Reserved	✔	✔	x	
11…13	FogSource	✔	✔	x	This field selects where and if the Fog message should be generated. D3D allows the user to provide a fog value in one of several places. The optioins are: 0 = None 1 = Specular alpha component 2 = Diffuse alpha component 3 = Front Specular Material alpha component 4 = Back Specular Material alpha component 5 = Colour alpha component

Notes: **MatrixMode** provides enables for BoundingBox and BoundingVolume tests and defines color order and fog source.

MatrixSelect

Name	Type	Offset	Format
MatrixSelect	Matrix *Command*	0xC680	integer

Bits	Name	Read	Write	Reset	Description
0..4	Option	✔	✔	x	This message holds the matrix stack to use for the various matrix commands. The options are: 0 = ModelView 1 = Projection 2 = Texture A 3 = Texture B 4 = Texture C 5 = Texture D 6 = Texture E 7 = Texture F 8 = Texture G 9 = Texture H 10 = View
5…31	Reserved	✘	✔	x	

MatrixStatus

Name	Type	Offset	Format
MatrixStatus	Matrix *Command*	0xC688	Bitfield

Bits	Name	Read	Write	Reset	Description
0,1	Reserved	✘	✔	x	
2	BoundingBoxOverrun	✘	✔	x	This bit, when set, indicates there has been an internal protocol error where the Command Unit asked the Matrix Unit to report the results of a bounding box test on a busy channel (i.e. the channel had its valid bit set). When this error occurs an interrupt is generated and this can be masked and polled via the normal PCI mechanisms.
3..31	Reserved	✘	✔	x	

Notes: Intended to clear the status results (any 3 bit value can be written), but mainly used for readback to detect if any errors have occurred and to return the matrix stack levels.
The error bits are sticky so will stay set until cleared by software and will attempt to generate an interrupt when an error occurs.

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.

ModelViewMatrix[16]

Name	Type	Offset	Format
ModelVie wMatrix[16]	Matrix *Control* Broadcast	0x9900	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Data word	✗	✔	x	

Notes: Loads one of the 16 **ModelView matrix** entries. The matrix is stored in column order so adjacent entries in the same column are at successive offsets.

ModelViewProjectionMatrix[16]

Name	Type	Offset	Format
ModelVie wProjection Matrix[16	Matrix *Control* Broadcast	0x9980	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Data word	✗	✔	x	

Notes: Loads one of the 16 **ModelViewProjectionmatrix** entries. The matrix is stored in column order so adjacent entries in the same column are at successive offsets.

NBFactor0
NBFactor1

Name	Type	Offset	Format
NBFacto 0	Normalization	0xC5F0	Float
NBFacto 1	Normalization	0xC5F8	Float
	Control		

Bits	Name	Read	Write	Reset	Description
0…31	BlendFactor	✔	✔	x	

Notes: Broadcast rewgister. These two messages hold the vertex blend factors to use when vertex blending. Before the incoming vertex normal (in the Normal register) is transformed and normalised it can be optionally blended with a second normal (provided in the **BlendNormal** register). The blending is done by the following vector equation:

v = Normal * NBFactor0 + BlendNormal * NBFactor1

where **NBFactor0** and **NBFactor1** are floating point scalars loaded by messages of the same name. No restrictions are places on the scalar values but typically they are configured with *alpha* and 1-*alpha* with 0 <= *alpha* <= 1 to simulate linear interpolation.

NormalizeMode
NormalizeModeAnd
NormalizeModeOr

Name	Type	Offset	Format
NormalizeMode	Normalize *Command* Broadcast	0x9518	Bitfield
NormalizeModeAnd	Normalize *Command* Broadcast	0xAAA0	Bitfield
NormalizeModeOr	Normalize *Command* Broadcast	0xAAA8	Bitfield

Bits	Name	Read	Write	Reset	Description
0	NormalEnable	✔	✔	x	When set causes any normals to be normalised. This operation is best avoided if possilbe as it is quite expensive.
1	reserved	✔	✔	x	
2	reserved	✔	✔	x	
3	AntialiasLine	✔	✔	x	When set causes the inverse line length to be calculated. This involves a square root so rather burden the Delta Unit with this operation it is done here instead.
4	NormaliseLight Position	✔	✔	x	This bit, when set, caused the incomming light's position to be normalised when it's W value is zero (i.e. the light position is really a direction).
5	NormaliseSpot Direction	✔	✔	x	This bit, when set, caused the incomming light's spot direction to be normalised.
6	BlendNormal	✔	✔	x	This bit, when set, enables the blending of normals before transformation.
7..31	Reserved				

Notes:

NormalMatrix[9]

Name	Type	Offset	Format
NormalMatrix[9]	Normalization *Control* Broadcast	0x9A00	Integer

Bits	Name	Read	Write	Reset	Description
0..3	Matrix	✔	✔	x	
4..31	Reserved	✘	✔	x	

Notes: Loads one of the 9 Normal matrix entries. The matrix is stored in column order so adjacent entries in the same column are at successive offsets. This matrix is passed on to the Normalisation Unit.

Nx
Ny
Nz

Name	Type	Offset	Format
Nx	Matrix	0x9810	Bitfield
Ny	Matrix	0x9808	Bitfield
Nz	Matrix	0x9800	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0..31	X word	✓	✓	x	Nx component of unpacked normal
32..63	Y word	✓	✓	x	Ny component of unpacked normal
64..95	Z word	✓	✓	x	Nz component of unpacked normal

Notes: This register holds the normal as received from the Command Unit. Its tag is changed to what the remaining units are expecting. The normal contains 3 data words for the x,y,z components.

ObjectIDValue

Name	Type	Offset	Format
ObjectID Value	PipeManager	0x95A8	Integer
	Control register		

Bits	Name	Read	Write	Reset	Description
0...15	ObjectID	✗	✓	x	Defines the value the ObjectID register is to take. Only the lower 16 bits are valid. Will send the ObjectID message to the rest of the system.
16..31	Unused	✓	✓	x	

Notes: Some antialiasing schemes work best when they know that triangles, etc. are really part of the same surface. This allows fragments along a common edge (from two triangles, for example) to be merged together, however this is only an optimisation and not fundamentally necessary for the antialiasing scheme to work.
ObjectIDs can be automatically generated on a Begin command and optionally for individual primitives when the Begin command indicates Points, Lines, Triangles or Quads as the primitive type. ObjectIDs can be controlled manually.

- The command **ObjectIDValue** is used to set the ObjectID (a 16 bit unsigned number) to the given value. It also informs the rasteriser about the new ObjectID value.
- The command **IncrementObjectID** increments the stored ObjectID value and then inform the rasteriser about the new ObjectID value. The ObjectID will roll over to zero when its range is exceeded.
- The current ObjectID value can be read back.

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
Packed4Pixels	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	

PackedBV…Packed V

Name	Type	Offset	Format
PackedB / Packed N / PackedFl / PackedT. A / PackedT. B / PackedT C / PackedT. D / PackedT. E / PackedT. F / PackedT G / PackedT. H / PackedV	Matrix	0xDA00...DA78	Integer
	Control		

Bits	Name	Read	Write	Reset	Description
0…31		✗	✔	x	Packed values

Notes: These messages hold the unsigned byte packed formats of the blend vertex, normal, face normal, texture coordinates A…H and vertex respectively.

PackedColor3

Name	Type	Offset	Format
PackedColor3	Matrix *Control register*	0x9898	Integer

Bits	Name	Read	Write	Reset	Description
0...31	PackedColor	✕	✓	x	Readback using **Color** tags

Notes: This message holds three 8 bit colour components in the least significant bytes. The missing alpha component is set to 1.0 when converted into the floating point internal format. The three bytes can hold RGB or BGR colour orders depending on the *ColourOrder* bit in **MatrixMode**.

PackedColor4

Name	Type	Offset	Format
PackedColor4	Matrix *Control register*	0x98A0	Integer

Bits	Name	Read	Write	Reset	Description
0...31	PackedColor	✕	✓	x	Readback using **Color** tags

Notes: This message holds four 8 bit colour components bytes. The four bytes can hold ARGB or ABGR colour orders depending on the *ColourOrder* bit in **MatrixMode**.

PackedDiffuse

Name	Type	Offset	Format
PackedDiffuse	Matrix *Control register* Broadcast	0x9CC8	Integer

Bits	Name	Read	Write	Reset	Description
0...31	DiffuseColor	✕	✓	x	

Notes: This message holds four 8 bit diffuse components in the least significant bytes. The four bytes can hold ARGB or ABGR colour orders depending on the *ColourOrder* bit in **MatrixMode.**

PackedNormal

Name	Type	Offset	Format
PackedNormal	Matrix *Control register* Broadcast	0x9CD0	Integer

Bits	Name	Read	Write	Reset	Description
0...31	Normal	✗	✓	x	

Notes: This message holds the normal in a compressed (or packed) format. The biased normal is packed into the 32 bit word as follows:
bits 0…9 = X
bits 10…19 = Y
bits 20…29 = Z
bits 30…31 = 0

PackedSpecular

Name	Type	Offset	Format
PackedSpecular	Matrix *Control register* Broadcast	0x9CC0	Integer

Bits	Name	Read	Write	Reset	Description
0...23	PackedColor	✗	✓	x	
24..31	Unused	✓	✓	x	

Notes: This message holds four 8 bit colour components bytes. The four bytes can hold ARGB or ABGR colour orders depending on the ColourOrder bit in MatrixMode. The biased normal is packed into the 32 bit word as follows:
bits 0…9 = X
bits 10…19 = Y
bits 20…29 = Z
bits 30…31 = 0

PassThrough

Name	Type	Offset	Format
PassThrough	Geometry *Control register*	0x8FF0	Integer

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

Notes: It can be useful to inject markers while in Feedback *RenderMode* (see the OpenGL function call *glPassThough* and the **GeometryMode** command) to help keep track of the which part of the model you are in. This is done by using the **PassThrough** command - the tag and data are written directly into the Host Out FIFO without changing any internal state in R5.

PCIWindowBase[0..7]

Name	Type	Offset	Format
PCIWind owBase[0..7]	Geometry *Control register*	0x8FF0	Integer

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

Notes: It can be useful to inject markers while in Feedback *RenderMode* (see the OpenGL function call *glPassThough* and the **GeometryMode** command) to help keep track of the which part of the model you are in. This is done by using the **PassThrough** command - the tag and data are written directly into the Host Out FIFO without changing any internal state in R5.

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.

PipeLoad
PipeLoadAnd
PipeLoadOr

Name	Type	Offset	Format
PipeLoad	PipeManager	0xC6E0	Bitfield
PipeLoad And	PipeManager	0xC610	Bitfield
PipeLoad Or	PipeManager	0xC618	Bitfield
	Command		

Bits	Name	Read	Write	Reset	Description
0	ForceReload	✕	✓	x	This bit, when set, forces all four vertex stores and the current values to be reloaded when a new pipe is started up. This should not normally be used and is only present at a test/debug aid.
1…7	TextureEnable	✕	✓	x	This 7 bit field holds one bit per multi texture (the 0th entry is always assumed set and so is omitted - this is the usual texture coordinate in OpenGL). Setting a bit enables the corresponding multi texture coordinate to be loaded when changing pipes. This is just an optimization and without it it would take an additional 14 cycles to change pipes.
8	FaceNormalEnable	✕	✓	x	This bit, when set, causes the current face normal to be loaded when changing pipes. This is just an optimization and without it it would take an additional 4 cycles to change pipes.
9	DiffuseEnable	✕	✓	x	This bit, when set, causes the current diffuse values to be loaded when changing pipes. This is just an optimization and without it it would take an additional 4 cycles to change pipes.
10	SpecularEnable	✕	✓	x	This bit, when set, causes the current specular values to be loaded when changing pipes. This is an optimization which saves 4 clock cycles per pipe change.
11	FogEnable	✕	✓	x	This bit, when set, causes the current fog value to be loaded when changing pipes. This is just an optimization and without it it would take an additional 4 cycles to change pipes.
12	BlendEnable	✕	✓	x	Flag
13…31	Reserved	✕	✓	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.

PipeMode
PipeModeAnd
PipeModeOr

Name	Type	Offset	Format
PipeMod	PipeManager	0x9560	Bitfield
PipeMod And	PipeManager	0xAB40	Bitfield
PipeMod Or	PipeManager	0xAB48	Bitfield
	Command broadcast		

Bits	Name	Read	Write	Reset	Description
0	UseOnePipe Only	✕	✓	x	This bit, when set, prevents the current pipe from changing based on the number of primitives processed or on the current pipe passing the busy threshold. It does not prevent raster pos, context processing or the SetPipe message from changing the current pipe. This is normally only used when the API enters some mode where multi-pipe operation doesn't work or is difficult. An example is the *SelectMode* for OpenGL.
1	SwitchPipe	✕	✓	x	This bit determines what condition is used to change the current pipe (assuming *UseOnePipeOnly* is false). The options are: 0 = PrimitiveCount 1 = PipeBusy PipeBusy is the normal mode and Primitive count is a test aid.
2	NextPipe	✕	✓	x	This bit determines how the next current pipe is chosen. The options are: 0 = RoundRobin 1 = LeastBusy LeastBusy is the normal mode and RoundRobin is a test aid.
3…7	MinPrimCount	✕	✓	x	This field holds the minimum number of primitives which must be processed by the current pipe before it can loose its current pipe status. This only has an effect when *SwitchPipe* is set to PipeBusy and *UseOnePipeOnly* is false.

8…14	MaxPrimCount	✕	✓	x	This field holds the maximum number of primitives which will be processed by the current pipe before it can loose its current pipe status. This only has an effect when SwitchPipe is set to PrimitiveCount and UseOnePipeOnly is false.
15…19	FifoBusyLevel	✕	✓	x	If the FIFO space becomes equal to or less than the amount in this field the FIFO is deemed to be 'busy' and is eligible to loose its current pipe status (also dependent on SwitchPipe and MinPrimCount fields).
20	ObjectTagEnable	✕	✓	x	When set causes the ObjectID to be incremented by the Begin message.
21	ObjectIDPerPrimitive	✕	✓	x	When set causes the ObjectID to be incremented on each point, line, triangle or quad when the primitive type is set to Points, Lines, Triangles or Quads. ObjectTagEnable must be set for this to happen.
22..31	reserved	✕	✓	x	

Notes: Controls the operation of the Pipe Manager unit. For example, the method of deciding when to change pipes can be made deterministic by setting the *SwitchPipe* bit to 0 which selects the number of canonical primitives each pipe is given before changing the a new pipe. The number of primitives to send into each pipe is held in the *MaxPrimCount* field. The selection of the next pipe can be made deterministic by setting the *NextPipe* bit to 0 which selects a round robin ordering of pipes.

PixelSize

Name	Type	Offset	Format
PixelSize	Rasterizer *Command*	0x80C0	Bitfield

Bits	Name	Read [32]	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

PointExtend

Name	Type	Offset	Format
PointExtend	Cull *Control register* Broadcast	0xC590	Fixed

Bits	Name	Read	Write	Reset	Description
0…11	Size	✓	✓	x	8.3 unsigned fixed point.
12…31	Reserved	0	0	x	

Notes: This register holds the number of scanlines to grow the y extent by when dealing with points. It is updated whenever the PointWidth value in the Delta Unit is updated. See also **AAPointExtend**

[32] 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.
1	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,when set, enables antialiasing of points qualified by the *AntialiasEnable* field in the **Begin** register. 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.

PolygonOffsetBias

Name	Type	Offset	Format
PolygonOffsetBias	Geometry *Control registers*	0x9BD0	float

Bits	Name	Read	Write	Reset	Description
0…31	Bias	✔	✔	x	

Notes: Holds the Polygon Offset Bias as a floating point number.

PolygonOffsetFactor

Name	Type	Offset	Format
PolygonOffsetFactor	Geometry *Control registers*	0x9B	float

Bits	Name	Read	Write	Reset	Description
0…31	Factor	✔	✔	x	

Notes: Polygon Offset Factor as a floating point number.

PopName

Name	Type	Offset	Format
PopNam	Geometry *Control registers*	0x95E8	float

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

Notes: Name stack manipulation. The name stack holds names (as 32 bit integers) the user can push, pop or load to keep track of the model hierarchy. The commands PushName, PopName, LoadName do these actions. PopName and LoadName update the stack with the 32 bit value in the data field. The name stack is reset with the **InitNames** command. The name stack is 64 entries deep. See also **SelectRecord** and the **FilterMode** *RenderMode* field..

PushName

Name	Type	Offset	Format
PushNam e	Geometry *Control registers*	0x95E0	float

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

Notes: Name stack manipulation. The name stack holds names (as 32 bit integers) the user can push, pop or load to keep track of the model hierarchy. The commands PushName, PopName, LoadName do these actions. PopName and LoadName update the stack with the 32 bit value in the data field. The name stack is reset with the **InitNames** command. The name stack is 64 entries deep. See also **SelectRecord** and the **FilterMode** *RenderMode* field..

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
RaasterizerMode	Rasterizer	0x80A0	Bitfield
RaasterizerModeAnd	Rasterizer	0xABA0	Bitfield
RaasterizerModeOr	Rasterizer	0xABA8	Bitfield
	Control register		

Bits	Name	Read [33]	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

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

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.
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.

RasterPosXIncrement

Name	Type	Offset	Format
RasterPosXIncrement	Geometry *Control register* Broadcast	0x9BE8	Float

Bits	Name	Read	Write	Reset	Description
0…31	Offset	✓	✓	x	Raster position Y offset

Notes: The amount to increment the raster position in X after a **GeomRectangle** command. The increment value is measured in pixels and is a floating point number.

RasterPosXOffset

Name	Type	Offset	Format
RasterPosXOffset	Geometry *Control register* Broadcast	0x9CE8	Float

Bits	Name	Read	Write	Reset	Description
0…31	Offset	✓	✓	x	Raster position Y offset

Notes: The amount to offset the raster position in X during a **GeomRectangle** command. The offset value is measured in pixels and is a floating point number. This does not update the raster position and is conditional on bit 2 of hte **GeomRectangle** command.

RasterPosYIncrement

Name	Type	Offset	Format
RasterPosYIncrement	Geometry *Control register* Broadcast	0x9BF0	Float

Bits	Name	Read	Write	Reset	Description
0…31	Offset	✓	✓	x	Raster position Y offset

Notes: The amount to offset the raster position in Y during a **GeomRectangle** command. The offset value is measured in pixels and is a floating point number. This does not update the raster position and is conditional on bit 2 of the **GeomRectangle** command.

RasterPosYOffset

Name	Type	Offset	Format
RasterPosYOffset	Geometry Control register Broadcast	0x8	Float

Bits	Name	Read	Write	Reset	Description
0…31	Offset	✓	✓	x	Raster position Y offset

otes: The amount to offset the raster position in Y during a **GeomRectangle** command. The offset value is measured in pixels and is a floating point number. This does not update the raster position and is conditional on bit 2 of hte **GeomRectangle** command.

ReadMonitorMode
ReadMonitorModeAnd
ReadMonitorModeOr

Name	Type	Offset	Format
ReadMonitorMode	ReadMonitor	0x80F8	Bitfield
ReadMonitorModeAnd	ReadMonitor	0xB5C0	Bitfield
ReadMonitorModeOr	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 R5s * 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.

Rectangle2DMode

Name	Type	Offset	Format
Rectangle2DMode	Delta *Control register*	0x	Float

Bits	Name	Read	Write	Reset	Description
0..31	RectangleMode	✓	✓	x	Holds the data used in the Render command sent to the rasterizer during ***GeomRectangle*** processing.

Notes: This register holds the width, height, fill direction and various enables used by the **DrawRectangle2D** command. See below for a full description.

Rectangle2DControl

Name	Type	Offset	Format
Rectangle2DControl	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 the rasterizer during ***GeomRectangle*** processing.

Notes: This register controls if window clipping is to be used for **DrawRectangle2D** command. When bit 0 is set window clipping is used so spans can not be used and local buffer reads are forced.

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:

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 the rasterizer during GeomRectangle 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.

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 register 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.
- This tag is not used by the geometry engine. It affects the rasterizer only and is retained for backwards compatibility

RepeatTriangle

Name	Type	Offset	Format
RepeatTr angle	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 a 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.
This tag is not used by the geometry engine. It affects the rasterizer only and is retained for backwards compatibility.

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. The data field is not used.

RestartPipe

Name	Type	Offset	Format
RestartPi e	Pipe Manager *Command*	0xC088	

Bits	Name	Read	Write	Reset	Description
0…31	Reserved	0	0	x	

Notes: After a context switc the **RestartPipe** command must be used to restart pipe processing. Pipe 0 is always the first pipe used after a context switch.

RestoreCurrent

Name	Type	Offset	Format
RestoreCurrent	Pipe Manager *Command*	0x95D0	Tag

Bits	Name	Read	Write	Reset	Description
0…31	Reserved	0	0	x	

Notes: This command restores the current normal, texture and colour values to be restored from the internal shadow registers.
OpenGL requires that the Evaluators light, shade and texture polygons without the current normal, texture and colour values being changed. This is achieved by saving them in internal registers, doing the evaluator operations and then restoring them. The commands to support this are **SaveCurrent** and **RestoreCurrent** respectively. This also incluses the 8 texture coordinates, face normal, diffuse, specular and fog values.

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.

RPx2

Name	Type	Offset	Format
RPx2	Command *Control register*	0x98E0	Wide Trigger

Bits	Name	Read	Write	Reset	Description
0..31	X word	✓	✓	x	Bounding vertex X coordinate
32..64	Y word	✓	✓	x	Bounding vertex Y coordinate

Notes: Holds the floating point value of the y, z and w components of the incoming raster position. The x component is supplied in one of the **RPx2, RPx3** or **RPx4** registers. The numerical postfix defines the which of the **RPy**, **RPz** registers should be used or have their default values used instead. **RPx2** generates the (x, y, 0.0, 1.0) vertex, **RPx3** generates the (x, y, z, 1.0) vertex and **RPx4** generates the (z, y, z, w) vertex. The **RPx**? register must be loaded after **RPy** and **RPz**.

RPx3

Name	Type	Offset	Format
RPx3	Command *Control register*	0x98E8	Wide Trigger

Bits	Name	Read	Write	Reset	Description
0..31	X	✓	✓	x	Bounding vertex X coordinate
32..63	Y	✓	✓	x	Bounding vertex Y coordinate
64..95	Z	✓	✓	x	Bounding vertex Z coordinate

Notes: Holds the floating point value of the y, z and w components of the incoming raster position. The x component is supplied in one of the **RPx2, RPx3** or **RPx4** registers. The numerical postfix defines the which of the **RPy**, **RPz** registers should be used or have their default values used instead. **RPx2** generates the (x, y, 0.0, 1.0) vertex, **RPx3** generates the (x, y, z, 1.0) vertex and **RPx4** generates the (z, y, z, w) vertex. The **RPx**? register must be loaded after **RPy** and **RPz**.

RPx4

Name	Type	Offset	Format
RPx4	Command *Control register*	0x98F0	Wide Trigger

Bits	Name	Read	Write	Reset	Description
0..31	X	✓	✓	x	Bounding vertex X coordinate
32..63	Y	✓	✓	x	Bounding vertex Y coordinate
64..95	Z	✓	✓	x	Bounding vertex Z coordinate
96..127	w	✓	✓	x	Bounding vertex W coordinate

Notes: Holds the floating point value of the y, z and w components of the incoming raster position. The x component is supplied in one of the **RPx2, RPx3** or **RPx4** registers. The numerical postfix defines the which of the **RPy**, **RPz** registers should be used or have their default values used instead. **RPx2** generates the (x, y, 0.0, 1.0) vertex, **RPx3** generates the (x, y, z, 1.0) vertex and **RPx4** generates the (z, y, z, w) vertex. The **RPx**? register must be loaded after **RPy** and **RPz**.

RPy

Name	Type	Offset	Format
RPy	Command *Control register*	0x98D8	Integer

Bits	Name	Read	Write	Reset	Description
0..31	Yvalue	✓	✓	x	Z component of blended vertex

Notes: The Y component of the incoming raster position supplied to build wide vertex data - see **RPx4**

RPz

Name	Type	Offset	Format
RPz	Matrix *Control register*	0x98D0	Integer

Bits	Name	Read	Write	Reset	Description
0..31	Zvalue	✓	✓	x	Z component of blended vertex

Notes: The Z component of the incoming raster position supplied to build wide vertex data - see **RPx4**

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.

SaveCurrent

Name	Type	Offset	Format
SaveCurr nt	Pipe Manager *Command*	0x95C8	Tag

Bits	Name	Read	Write	Reset	Description
0…31	Reserved	0	0	x	

Notes: This command saves the current normal, texture and colour values to be restored from the internal shadow registers.
OpenGL requires that the Evaluators light, shade and texture polygons without the current normal, texture and colour values being changed. This is achieved by saving them in internal registers, doing the evaluator operations and then restoring them. The commands to support this are **SaveCurrent** and **RestoreCurrent** respectively. This also incluses the 8 texture coordinates, face normal, diffuse, specular and fog 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 register defines which R5 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 R5 in the system.

SceneAmbientColorBlue

Name	Type	Offset	Format
SceneAmbientColorBlue	Material *Control register* Broadcast	0xA810	Float

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

Notes: Ambient blue scene color in floating point format

SceneAmbientColorGreen

Name	Type	Offset	Format
SceneAmbientColorGreen	Material *Control register* Broadcast	0xA808	Float

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

Notes: Ambient green scene color in floating point format

SceneAmbientColorRed

Name	Type	Offset	Format
SceneAmbientColorRed	Material *Control register* Broadcast	0xA800	Float

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

Notes: Ambient red scene color in floating point format

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 [34]	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.

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

Scratch

Name	Type	Offset	Format
Scratch	Command *Control register*	0x	Bitfield

Bits	Name	Read	Write	Reset	Description
0…31	Data	✓	✓	x	User-defined data

Notes:

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: Security filters out any commands that may cause the pipeline to lockup if used incorrectly. If the security mode is enabled, potentially dangerous registers can only be programmed by a direct write, not through a packet. This stops DMA buffers in user address space that are corrupted by another application causing the chip to lockup.
The following commands are filtered out of packets and discarded if the security bit is set:

- Security
- ContextDump
- ContextRestore
- ContextData
- TMask
- DataSync
- CommandStatus[0..15]

SelectRecord

Name	Type	Offset	Format
SelectRec ord	Geometry *Control register*	0x8FE8	Bitfield

Bits	Name	Read	Write	Reset	Description
0…31	Data	✓	✓	x	User-defined data

Notes: Holds the result of the Select operation. It will hold the name stack depth, min and max Z values and the name stack contents at various times.
If the hit flag is set when a name stack manipulation is done a hit record is written to the rasterizer host output FIFO. The hit flag is then reset along with the minimum and maximum Z range. The hit record consists of (in order):

1. The count of the names on the stack (+ some error flags),
2. The minimum Z value as a normalised floating point number,
3. The maximum Z value as a normalised floating point number,
4. The name stack entries, oldest first (variable number [0…64] of words.

Bits 14 and 15 in the **FilterMode** register must be Set to allow the **SelectRecord** tag and data values to be written in to the FIFO. The name stack manipulations commands are ignored when not in Select mode.

The hit record data requires a minimum of modification to be fully OpenGL compliant. The software needs to parse the hit record and convert the minimum and maximum Z values from the floating point format (normalised to be in the range 0.0…1.0) to the 32 bit integer format required by OpenGL.

The *NameCount* value has the following fields:

Bit	Name	Description
0…6	Count	This field holds the number of names on the name stack.
7…28		Not used.
29	InvalidOperation	A LoadName operation was attempted on an empty name stack when this hit record was being collected. This is cleared for subsequent hit records (unless they manifest this error) however the stack may no longer be totally valid.
30	StackUnderflow	The name stack was popped while empty when this hit record was being collected. This is cleared for subsequent hit records (unless they manifest this error) however the stack may no longer be totally valid.
31	StackOverflow	The name stack was pushed while full when this hit record was being collected. This is cleared for subsequent hit records (unless they manifest this error) however the stack may no longer be totally valid.

During SelectMode operation the lighting is still calculated even though it is never used so a useful optimisation is to disable lighting. Similarly for texture and fog modes. Also it is advisable to disable short line and small triangle threshold testing and always do a full clip to avoid selecting primitives which would have been clipped out.

SelectResult

Name	Type	Offset	Format
SelectResult	Geometry *Command register*	0x9580	Tag

Bits	Name	Read	Write	Reset	Description
0…31	Reserved	✔	✔	x	

Notes: This command writes the current hit record into the message stream if a hit has occured. The Hit flag is then reset. See **SelectRecord** for the structure of the Hit record.

SetDeltaPort

Name	Type	Offset	Format
SetDeltaPort	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 register 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.

SetLightPipe

Name	Type	Offset	Format
SetLightPipe	LightMux Unit *Command register*	0x96F8	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Port number	✘	✔	x	Port

Notes: This command sets which Light port should be made the current port and receive subsequent data.

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.

SetPipe

Name	Type	Offset	Format
SetPipe	Pipe Manager *Command register*	0xC080	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Port number	✕	✔	x	Port

Notes: Writing the **SetPipe** command to the current pipe with the new pipe number in the data field directs the Pipe Mux Unit to stop taking messages from the current pipe and change to the new pipe. The new pipe then becomes the current pipe. Once the new pipe has been established as the current pipe the vertex stores and current values are re-instated using the **RestoreCurrent** command.
SetPipe is used for testing and should not be needed to support the allocation of tasks to pipes during normal operation.

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.

StartBoundingVolume

Name	Type	Offset	Format
StartBoundingVolume	Matrix *Command register*	0xC880	Tag

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

Notes: This command prepares for a bounding volume test by resetting the outcodes and initialising any matrices. The bounding volume (normally a convex hull) is defined a vertex at a time using the BoundingVertexX, BoundingVertexY and BoundingVertexZ messages.
The **BoundingVertexZ** register must be entered last as it triggers the bounding vertex to be added to the volume under test.
The bounding volume is terminated using the **EndBoundingVolume** command which directs the command unit to prompt for the result. The bounding volume test is enabled by the *BoundingVolumeEnable* bit in the **MatrixMode** message. If the test is disabled then it always returns the bounding box as being 'in view'.

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 *Command*	0xAD18	Bitfield Logic Mask

Bits	Name	Read [35]	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.

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

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.

Stencil

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

Bits	Name	Read	Write	Reset	Description
0…7	Stencil value	✔	✔	x	8 bit stencil value
8…31	Reserved	0	0	x	

Notes: The ***Stencil*** register holds an externally sourced stencil value. It is a 32 bit register of which only the least significant 8 bits are used. The unused most significant bits should be set to zero. Set the register to the 8 bit stencil value to be used in clearing down the stencil buffer, or in drawing a primitive where the host supplies the stencil value.

StencilData
StencilDataAnd
StencilDataOr

Name	Type	Offset	Format
StencilData	Stencil	0x8990	Bitfield
StencilDataAnd	Stencil	0xB3E0	Bitfield Logic Mask
StencilDataOr	Stencil	0xB3E8	Bitfield Logic Mask

Control registers

Bits	Name	Read [36]	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.

[36] 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 [37]	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.

[37] 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

StripeFilterMode
StripeFilterModeAnd
StripeFilterModeOr

Name	Type	Offset	Format
StripeFilterMode	Stencil	0x8988	Bitfield
StripeFilterModeAnd	Stencil	0xAC60	Bitfield Logic Mask
StripeFilterModeOr	Stencil	0xAC68	Bitfield Logic Mask

Command registers

Bits	Name	Read [38]	Write	Reset	Description
0	FilterPoints	✔	✔	x	This bit is set whenever aliased points are to be included in stripe filtering (the points may originate from a triangle when the polymode is set to points). The PointExtend message holds the 'width' of the point (see later).
1	FilterAAPoints	✔	✔	x	This bit is set whenever antialiased points are to be included in stripe filtering (the points may originate from a triangle when the polymode is set to points). The AAPointExtend message holds the 'width' of the point (see later).

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

2	AAPointEnable	✔	✔	x	This bit is set when points are to be drawn antialiased. It would normally track the AntialiasEnable bit in the PointMode message. This is further qualified by the AntialiasEnable bit in the Begin message.
3	AAPoint Quality	✔	✔	x	This bit determines the antialiasing quality to apply to antialiased points. The two options are: 0 = 4x4 1 = 8x8 This would normally track the AntialiasingQuality field in the PointMode message.
4	FilterLines	✔	✔	x	This bit is set whenever aliased lines are to be included in stripe filtering (the lines may originate from a triangle when the polymode is set to lines). The LineExtend message holds the 'width' of the line (see later).
5	FilterAALines	✔	✔	x	This bit is set whenever antialiased lines are to be included in stripe filtering (the lines may originate from a triangle when the polymode is set to lines). The AALineExtend message holds the 'width' of the line (see later).
6	AALineEnable	✔	✔	x	This bit is set when lines are to be drawn antialiased. It would normally track the AntialiasEnable bit in the LineMode message. This is further qualified by the AntialiasEnable bit in the Begin message.
7	AALineQuality	✔	✔	x	This bit determines the antialiasing quality to apply to antialiased lines. The two options are: 0 = 4x4 1 = 8x8 This would normally track the AntialiasingQuality field in the LineMode message.
8	FilterTriangles	✔	✔	x	This bit is set whenever aliased triangles are to be included in stripe filtering. The TriangleExtend message holds any tolerance to apply to the test (see later).
9	FilterAA Triangles	✔	✔	x	This bit is set whenever antialiased triangles are to be included in stripe filtering. The AATriangleExtend message holds any tolerance to apply to the test (see later).
10	Enable	✔	✔	x	This bit is set when triangles are to be drawn antialiased. It would normally track the AntialiasEnable bit in the TriangleMode message. This is further qualified by the AntialiasEnable bit in the Begin message.
11	AATriangle Quality	✔	✔	x	This bit determines the antialiasing quality to apply to antialiased triangles. The two options are: 0 = 4x4 1 = 8x8 This would normally track the AntialiasingQuality field in the TriangleMode message.

12	RejectZero HeightTriangles	✔	✔	x	This bit, when set, allows triangles with a zero height to be rejected. This is a desirable operation, however it can be disabled just in case any numerical differences occur in the calculation (vs. what the Delta Unit would determine) which give a false positive result.
13	Bias Coordinates	✔	✔	x	This bit, when set, will remove any coordinate biasing in Y before the stripe ownership test is done. This will only be set if the viewport biasing includes the bias and the Delta Unit is enabled to remove the bias. The bias is held as a floating point number in the StripeYBias message.
14…16	StripeHeight	0	0	x	This field specifies the number of scanlines in a stripe. The options are: 0 = 1 3 = 8 6 = 64 1 = 2 4 = 16 7 = 128 2 = 4 5 = 32
17…18	NumberRasterisers	✔	✔	x	This field specifies the number of rasterisers. The options are: 0 = 1 1 = 2 2 = 4 3 = 8
19..31	Reserved	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.

StripeFilterYOffset

Name	Type	Offset	Format
StripeFilt rYOffset	Cull Unit *Control register*	0xC588	Fixed point

Bits	Name	Read	Write	Reset	Description
0…15	Y offset	✓	✓	x	2's complement fixed point value
16..31	Reserved	✗	✓	x	

Notes: This message hold the y offset value (16 bit 2's complement) used to position the window on the screen (all the stripe calculations are done relative to the origin of the screen and not relative to the origin of the window). It should track the *StripeOffsetY* value loaded into the rasteriser.

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.

StripeOwnership

Name	Type	Offset	Format
StripeOwnership	*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 message holds information which identified which stripes this Gamma owns. The fields are:
bit 0 = Stripe0Enable
bit 1…3 = Stripe0
bit 4 = Stripe1Enable
bit 5…7 = Stripe1
If the enable bit is 0 then this stripe is not present (maybe because no rasteriser is connected to this port). When an enable bit is set the corresponding stripe field is set to 0…7 to indicate which stripe number the rasteriser owns.

StripeYBias

Name	Type	Offset	Format
StripeYBias	*Control register*	0x80C8	Float

Bits	Name	Read	Write	Reset	Description
0…15	Y bias	✓	✓	x	
16…23	Reserved	0	0	x	

Notes: This message holds the Y bias value applied during geometry processing to improve floating point accuracy. This value should track the YBias value loaded into the Delta Unit. It is a floating point number.

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 the rasterizer. 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
Sync	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.

TAq

Name	Type	Offset	Format
TAq	Matrix *Control register*	0xC380	Float

Bits	Name	Read	Write	Reset	Description
0..31	Qvalue	✕	✓	x	Q component of incoming Texture A

Notes: The Q component of wide vertex data - see **TAs1**

TAr

Name	Type	Offset	Format
TAr	Matrix *Control register*	0xC388	Float

Bits	Name	Read	Write	Reset	Description
0..31	Rvalue	✕	✓	x	R component of incoming Texture A

Notes: The R component of wide vertex data - see **TAs1**

TAs1
TAs2
TAs3
TAs3q
TAs4

Name	Type	Offset	Format
TAs1	Matrix	0xC398	Wide
TAs2	Matrix	0xC3A0	Wide
TAs3	Matrix	0xC3A8	Wide
TAs3q	Matrix	0xC3B8	Wide
TAs4	Matrix	0xC3B0	Wide
	Control register		

Bits	Name	Read [39]	Write	Reset	Description
0..31	S	✗	✔	x	Texture A co-ordinate S component
32..63	T	✗	✔	x	Texture A co-ordinate T component
64..95	R	✗	✔	x	Texture A co-ordinate R component
96..127	Q	✗	✔	x	Texture A co-ordinate Q component

Notes: These registers hold the incoming texture A coordinate, and the coordinate has 1 (s), 2 (s, t), 3 (s, t, r) or 4 (s, t, r, q) components. The missing t and r components are replaced with 0 and the missing q by 1. The register is used on output. **TAs3q** has r missing and not q as might be expected

TAt

Name	Type	Offset	Format
TAt	Matrix *Control register*	0xC390	Float

Bits	Name	Read	Write	Reset	Description
0..31	Tvalue	✗	✓	x	T component of incoming Texture A

Notes: The T component of wide vertex data - see **TAs1**

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

TBq

Name	Type	Offset	Format
TBq	Matrix *Control register*	0xC3C0	Float

Bits	Name	Read	Write	Reset	Description
0..31	Qvalue	✕	✓	x	Q component of incoming Texture B

Notes: The Q component of wide vertex data - see **TBs1**

TBr

Name	Type	Offset	Format
TBr	Matrix *Control register*	0xC3C8	Float

Bits	Name	Read	Write	Reset	Description
0..31	Rvalue	✕	✓	x	R component of incoming Texture B

Notes: The R component supplied to build wide vertex data - see **TBs1**

TBs1
TBs2
TBs3
TBs3q
TBs4

Name	Type	Offset	Format
TBs1	Matrix	0xC3D8	Wide
TBs2	Matrix	0xC3E0	Wide
TBs3	Matrix	0xC3E8	Wide
TBs3q	Matrix	0xC3F8	Wide
TBs4	Matrix	0xC3F0	Wide
	Control register		

Bits	Name	Read [40]	Write	Reset	Description
0..31	S	✗	✔	x	Texture B co-ordinate S component
32..63	T	✗	✔	x	Texture B co-ordinate T component
64..95	R	✗	✔	x	Texture B co-ordinate R component
96..127	Q	✗	✔	x	Texture B co-ordinate Q component

Notes: These registers hold the incoming texture B coordinate, and the coordinate has 1 (s), 2 (s, t), 3 (s, t, r) or 4 (s, t, r, q) components. The missing t and r components are replaced with 0 and the missing q by 1. The register is used on output. **TBs3q** has r missing and not q as might be expected

TBt

Name	Type	Offset	Format
TBt	Matrix	0xC3D0	Float
	Control register		

Bits	Name	Read	Write	Reset	Description
0..31	Tvalue	✗	✓	x	T component of incoming Texture B

Notes: The T component of wide vertex data - see **TBs1**

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

TCq

Name	Type	Offset	Format
TCq	Matrix *Control register*	0xC400	Float

Bits	Name	Read	Write	Reset	Description
0..31	Qvalue	✕	✓	x	Q component of incoming Texture C

Notes: The Q component of wide vertex data - see **TCs1**

TCr

Name	Type	Offset	Format
TCr	Matrix *Control register*	0xC408	Float

Bits	Name	Read	Write	Reset	Description
0..31	Rvalue	✕	✓	x	R component of incoming Texture C

Notes: The R component of wide vertex data - see **TCs1**

TCs1
TCs2
TCs3
TCs3q
TCs4

Name	Type	Offset	Format
TCs1	Matrix	0xC410	Wide
TCs2	Matrix	0xC420	Wide
TCs3	Matrix	0xC428	Wide
TCs3q	Matrix	0xC438	Wide
TCs4	Matrix	0xC430	Wide
	Control register		

Bits	Name	Read [41]	Write	Reset	Description
0..31	S	✗	✔	x	Texture C co-ordinate S component
32..63	T	✗	✔	x	Texture C co-ordinate T component
64..95	R	✗	✔	x	Texture C co-ordinate R component
96..127	Q	✗	✔	x	Texture C co-ordinate Q component

Notes: These registers hold the incoming texture C coordinate, and the coordinate has 1 (s), 2 (s, t), 3 (s, t, r) or 4 (s, t, r, q) components. The missing t and r components are replaced with 0 and the missing q by 1. The register is used on output. **TCs3q** has r missing and not q as might be expected

TCt

Name	Type	Offset	Format
TCt	Matrix	0xC410	Float
	Control register		

Bits	Name	Read	Write	Reset	Description
0..31	Tvalue	✗	✓	x	T component of incoming Texture C

Notes: The T component supplied to build wide vertex data - see **TCs1**

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

TDq

Name	Type	Offset	Format
TDq	Matrix *Control register*	0xC440	Float

Bits	Name	Read	Write	Reset	Description
0..31	Qvalue	✕	✓	x	Q component of incoming Texture A

Notes: The Q component supplied to build wide vertex data - see **TDs1**

TDr

Name	Type	Offset	Format
TDr	Matrix *Control register*	0xC448	Float

Bits	Name	Read	Write	Reset	Description
0..31	Rvalue	✕	✓	x	R component of incoming Texture D

Notes: The R component supplied to build wide vertex data - see **TDs1**

TDs1
TDs2
TDs3
TDs3q
TDs4

Name	Type	Offset	Format
TDs1	Matrix	0xC458	Wide
TDs2	Matrix	0xC460	Wide
TDs3	Matrix	0xC468	Wide
TDs3q	Matrix	0xC478	Wide
TDs4	Matrix	0xC470	Wide
	Control register		

Bits	Name	Read [42]	Write	Reset	Description
0..31	S	✗	✔	x	Texture D co-ordinate S component
32..63	T	✗	✔	x	Texture D co-ordinate T component
64..95	R	✗	✔	x	Texture D co-ordinate R component
96..127	Q	✗	✔	x	Texture D co-ordinate Q component

Notes: These registers hold the incoming texture D coordinate, and the coordinate has 1 (s), 2 (s, t), 3 (s, t, r) or 4 (s, t, r, q) components. The missing t and r components are replaced with 0 and the missing q by 1. The register is used on output. **TDs3q** has r missing and not q as might be expected

TDt

Name	Type	Offset	Format
TDt	Matrix	0xC450	Float
	Control register		

Bits	Name	Read	Write	Reset	Description
0..31	Tvalue	✗	✓	x	T component of incoming Texture D

Notes: The T component supplied to build wide vertex data - see **TDs1**

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

TEq

Name	Type	Offset	Format
Teq	Matrix *Control register*	0xC400	Float

Bits	Name	Read	Write	Reset	Description
0..31	Qvalue	✕	✓	x	Q component of incoming Texture E

Notes: The Q component supplied to build wide vertex data - see **TEs1**

TEr

Name	Type	Offset	Format
TEr	Matrix *Control register*	0xC488	Float

Bits	Name	Read	Write	Reset	Description
0..31	Rvalue	✕	✓	x	R component of incoming Texture E

Notes: The R component supplied to build wide vertex data - see **TEs1**

TEs1
TEs2
TEs3
TEs3q
TEs4

Name	Type	Offset	Format
TEs1	Matrix	0xC498	Wide
TEs2	Matrix	0xC4A0	Wide
TEs3	Matrix	0xC4A8	Wide
TEs3q	Matrix	0xC4B8	Wide
TEs4	Matrix	0xC4B0	Wide
	Control register		

Bits	Name	Read [43]	Write	Reset	Description
0..31	S	✗	✔	x	Texture E co-ordinate S component
32..63	T	✗	✔	x	Texture E co-ordinate T component
64..95	R	✗	✔	x	Texture E co-ordinate R component
96..127	Q	✗	✔	x	Texture E co-ordinate Q component

Notes: These registers hold the incoming texture E coordinate, and the coordinate has 1 (s), 2 (s, t), 3 (s, t, r) or 4 (s, t, r, q) components. The missing t and r components are replaced with 0 and the missing q by 1. The register is used on output. **TEs3q** has r missing and not q as might be expected

TEt

Name	Type	Offset	Format
TEt	Matrix	0xC490	Float
	Control register		

Bits	Name	Read	Write	Reset	Description
0..31	Tvalue	✗	✓	x	T component of incoming Texture E

Notes: The T component supplied to build wide vertex data - see **TEs1**

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

TexGen[0..15]

Name	Type	Offset	Format
TexGen[0..15]	Texture Fog *Control register*	0x9B00-0x9B78	Float

Bits	Name	Read	Write	Reset	Description
0..31	TexGen Coefficient	✓	✓	x	

Notes: The registers hold the texture generation coefficients to be used for object and eye linear texture generation. Each texture component has 4 associated registers to hold the coefficients for object- or eye-linear operations for that component:

S Texture Component	TexGen0..3
T Texture Component	TexGen4..7
R Texture Component	TexGen8..11
Q Texture Component	TexGen12..15

The coefficients are stored in column order so adjacent entries in the same column are at successive offsets. The TextureTexGenSelect register identifies which one of the 8 sets of texture generation coefficients will be modified. The texture generation modes are controlled by the **TransformMode** register

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 *Command*	0x8778	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: 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 [44]	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)

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

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.
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	0x8500	Integer

Rasterizer Control register

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

Notes: This register holds the base address of each texture map (or level for a mip map). The address should be aligned to the natural size of the texture map, however some layouts impose additional restrictions. The *MapBaseRegister* field of the **TextureReadMode** register defines which **TextureBaseAddr** register should be used to hold the address for map level 0 when mip mapping, or the texture map when not mip mapping. Successive map levels are at increasing **TextureBaseAddr** registers upto (and including) the MapMaxLevel. 3D textures always use **TextureBaseAddr0**.

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 = Color.A 2 = TextureCompositeFactor0.A 3 = Texel0.A 4 = Texel1.A where C is the RGB or A depending on the channel.
14	InvertI	✔	✔	x	This bit, if set, will invert the selected I value before it is used.
15	A	✔	✔	x	This bit selects which Arg (after any inversion) is to be used as A in the Operation. The options are: 0 = Arg1 1 = Arg2
16	B	✔	✔	x	This bit selects which Arg (after any inversion) is to be used as B in the Operation. The options are: 0 = Arg1 1 = Arg2

17…20	Operation	✔	✔	x	This field defines how the three inputs (A, B and I) are combined. Note the inputs can be optionally inverted before being combined. The 8 bit inputs are unsigned 0.8 fixed point format, but 255 is treated as if it were 1.0 for the calculations. The possible operations are: 0 = Pass (A) 1 = Add (A + B) 2 = AddSigned (A + B - 128) 3 = Subtract (A - B) 4 = Modulate (A * B) 5 = Lerp (A * (1.0 - I) + B * I) 6 = ModulateColorAddAlpha (A * B + I) 7 = ModulateAlphaAddColor (A * I + B) 8 = AddSmoothSaturate (A + B - A * B) 9 = ModulateSigned (A * B, but A and B are biased 8 bit numbers)
21…22	Scale	0	0	x	This field selects the scale factor to apply to the final result before it is clamped. The options are: 0 = 0.5 1 = 1 2 = 2 3 = 4
23…31	Reserved	0	0	x	

Notes: The Texture unit composites the Color, Texel0 and Texel1 fragment's values with one or two constant color values held in registers and passes the result on to the next unit as a texture value.
The compositing is done in two stages and is controlled separately for the RGB channels and the Alpha channel. This register defines the operation for the alpha channels in compositing stage 0 for this unit.

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

TextureCompositeAlphaMode1
TextureCompositeAlphaMode1And
TextureCompositeAlphaMode1Or

Name	Type	Offset	Format
TextureCompositeAlphaMode1	Texture	0xB320	Bitfield
TextureCompositeAlphaMode1And	Texture	0xB3B0	Bitfield Logic Mask
TextureCompositeAlphaMode1Or	Texture	0xB3B8	Bitfield Logic Mask

Control registers

Bits	Name	Read [45]	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.

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

6…9	Arg2	✔	✔	x	This field selects the source value for Arg2. The options are: 0 = Output.C of the previous stage or height if the first stage 1 = Output.A of the previous stage or height if the first stage 2 = Color.C 3 = Color.A 4 = TextureCompositeFactor1C 5 = TextureCompositeFactor1A 6 = Texel0.C 7 = Texel0.A 8 = Texel1.C 9 = Texel1.A 10 = Sum of the color components of the previous stage or 0 if the first stage. where C is the RGB or A depending on the channel. height is defined as clamp (Texel0.A - Texel1.A + 128)
10	InvertArg2	✔	✔	x	This bit, if set, will invert the selected Arg2 value before it is used.
11…13	I	✔	✔	x	This field selects what is used as the interpolation factor when the Operation field is set to Lerp, for example. The options are: 0 = Output.A of the previous stage or 0 if the first stage 1 = Color.A 2 = TextureCompositeFactor1.A 3 = Texel0.A 4 = Texel1.A where C is the RGB or A depending on the channel.
14	InvertI	✔	✔	x	This bit, if set, will invert the selected I value before it is used.
15	A	✔	✔	x	This bit selects which Arg (after any inversion) is to be used as A in the Operation. The options are: 0 = Arg1 1 = Arg2
16	B	✔	✔	x	This bit selects which Arg (after any inversion) is to be used as B in the Operation. The options are: 0 = Arg1 1 = Arg2

17…20	Operation	✔	✔	x	This field defines how the three inputs (A, B and I) are combined. Note the inputs can be optionally inverted before being combined. The 8 bit inputs are unsigned 0.8 fixed point format, but 255 is treated as if it were 1.0 for the calculations. The possible operations are: 0 = Pass (A) 1 = Add (A + B) 2 = AddSigned (A + B - 128) 3 = Subtract (A - B) 4 = Modulate (A * B) 5 = Lerp (A * (1.0 - I) + B * I) 6 = ModulateColorAddAlpha (A * B + I) 7 = ModulateAlphaAddColor (A * I + B) 8 = AddSmoothSaturate (A + B - A * B) 9 = ModulateSigned (A * B, but A and B are biased 8 bit numbers)
21…22	Scale	✔	✔	x	This field selects the scale factor to apply to the final result before it is clamped. The options are: 0 = 0.5 1 = 1 2 = 2 3 = 4
23…31	Reserved	0	0	x	

Notes: The Texture unit composites the fragment's Color, Texel0 and Texel1 values with one or two constant color values held in registers and passes the result on to the next unit as a texture value.
The compositing is done in two stages and is controlled separately for the RGB channels and the Alpha channel. This register defines the operation for the alpha channels in compositing stage 0 for this unit.

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

TextureCompositeColorMode0
TextureCompositeColorMode0And
TextureCompositeColorMode0Or

Name	Type	Offset	Format
TextureCompositeColorMode0	Texture	0xB308	Bitfield
TextureCompositeColorMode0And	Texture	0xB380	Bitfield Logic Mask
TextureCompositeColorMode0Or	Texture	0xB388	Bitfield Logic Mask

Control registers

Bits	Name	Read	Write	Reset	Description
0	Enable	✔	✔	x	When set causes the output to be calculated as defined by the fields in this register, otherwise the texel0 data is passed through for stage0 and Output data is passed through for stage 1.
1…4	Arg1	✔	✔	x	This field selects the source value for Arg1. The options are: 0 = Output.C of the previous stage or height if the first stage 1 = Output.A of the previous stage or height if the first stage 2 = Color.C 3 = Color.A 4 = TextureCompositeFactor0.C 5 = TextureCompositeFactor0.A 6 = Texel0.C 7 = Texel0.A 8 = Texel1.C 9 = Texel1.A 10 = Sum of the color components of the previous stage or 0 if the first stage. where C is the RGB or A depending on the channel. Height is defined as clamp (Texel0.A - Texel1.A + 128)
5	InvertArg1	✔	✔	x	This bit, if set, will invert the selected Arg1 value before it is used.

6…9	Arg2	✔	✔	x	This field selects the source value for Arg2. The options are: 0 = Output.C of the previous stage or height if the first stage 1 = Output.A of the previous stage or height if the first stage 2 = Color.C 3 = Color.A 4 = TextureCompositeFactor0.C 5 = TextureCompositeFactor0.A 6 = Texel0.C 7 = Texel0.A 8 = Texel1.C 9 = Texel1.A 10 = Sum of the color components of the previous stage or 0 if the first stage. where C is the RGB or A depending on the channel. height is defined as clamp (Texel0.A - Texel1.A + 128)
10	InvertArg2	✔	✔	x	This bit, if set, will invert the selected Arg2 value before it is used.
11…13	I	✔	✔	x	This field selects what is used as the interpolation factor when the Operation field is set to Lerp, for example. The options are: 0 = Output.A of the previous stage or 0 if the first stage 1 = Color.A 2 = TextureCompositeFactor*0*.A 3 = Texel0.A 4 = Texel1.A where C is the RGB or A depending on the channel.
14	InvertI	✔	✔	x	This bit, if set, will invert the selected I value before it is used.
15	A	✔	✔	x	This bit selects which Arg (after any inversion) is to be used as A in the Operation. The options are: 0 = Arg1 1 = Arg2
16	B	✔	✔	x	This bit selects which Arg (after any inversion) is to be used as B in the Operation. The options are: 0 = Arg1 1 = Arg2

17…20	Operation	✔	✔	x	This field defines how the three inputs (A, B and I) are combined. Note the inputs can be optionally inverted before being combined. The 8 bit inputs are unsigned 0.8 fixed point format, but 255 is treated as if it were 1.0 for the calculations. The possible operations are: 0 = Pass (A) 1 = Add (A + B) 2 = AddSigned (A + B - 128) 3 = Subtract (A - B) 4 = Modulate (A * B) 5 = Lerp (A * (1.0 - I) + B * I) 6 = ModulateColorAddAlpha (A * B + I) 7 = ModulateAlphaAddColor (A * I + B) 8 = AddSmoothSaturate (A + B - A * B) 9 = ModulateSigned (A * B, but A and B are biased 8 bit numbers)
21…22	Scale	✔	✔	x	This field selects the scale factor to apply to the final result before it is clamped. The options are: 0 = 0.5 1 = 1 2 = 2 3 = 4
23…31	Reserved	0	0	x	

Notes: The Texture unit composites the framgent's Color, Texel0 and Texel1 values with one or two constant color values held in registers and passes the result on to the next unit as a texture value.
The compositing is done in two stages and is controlled separately for the RGB channels and the Alpha channel. This register defines the operation for the alpha channels in compositing stage 0 for this unit.

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

TextureCompositeColorMode1
TextureCompositeColorMode1And
TextureCompositeColorMode1Or

Name	Type	Offset	Format
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 register suffix and C is the RGB or A depending on the channel. height is defined as clamp (Texel0.A - Texel1.A + 128)
5	InvertArg1	✔	✔	x	This bit, if set, will invert the selected Arg1 value before it is used.

6…9	Arg2	✔	✔	x	This field selects the source value for Arg2. The options are: 0 = Output.C of the previous stage or height if the first stage 1 = Output.A of the previous stage or height if the first stage 2 = Color.C 3 = Color.A 4 = TextureCompositeFactor1.C 5 = TextureCompositeFactor1.A 6 = Texel0.C 7 = Texel0.A 8 = Texel1.C 9 = Texel1.A 10 = Sum of the color components of the previous stage or 0 if the first stage. where C is the RGB or A depending on the channel. height is defined as clamp (Texel0.A - Texel1.A + 128)
10	InvertArg2	✔	✔	x	This bit, if set, will invert the selected Arg2 value before it is used.
11…13	I	✔	✔	x	This field selects what is used as the interpolation factor when the Operation field is set to Lerp, for example. The options are: 0 = Output.A of the previous stage or 0 if the first stage 1 = Color.A 2 = TextureCompositeFactor1.A 3 = Texel0.A 4 = Texel1.A where C is the RGB or A depending on the channel.
14	InvertI	✔	✔	x	This bit, if set, will invert the selected I value before it is used.
15	A	✔	✔	x	This bit selects which Arg (after any inversion) is to be used as A in the Operation. The options are: 0 = Arg1 1 = Arg2
16	B	✔	✔	x	This bit selects which Arg (after any inversion) is to be used as B in the Operation. The options are: 0 = Arg1 1 = Arg2

17…20	Operation	✔	✔	x	This field defines how the three inputs (A, B and I) are combined. Note the inputs can be optionally inverted before being combined. The 8 bit inputs are unsigned 0.8 fixed point format, but 255 is treated as if it were 1.0 for the calculations. The possible operations are: 0 = Pass (A) 1 = Add (A + B) 2 = AddSigned (A + B - 128) 3 = Subtract (A - B) 4 = Modulate (A * B) 5 = Lerp (A * (1.0 - I) + B * I) 6 = ModulateColorAddAlpha (A * B + I) 7 = ModulateAlphaAddColor (A * I + B) 8 = AddSmoothSaturate (A + B - A * B) 9 = ModulateSigned (A * B, but A and B are biased 8 bit numbers)
21…22	Scale	✔	✔	x	This field selects the scale factor to apply to the final result before it is clamped. The options are: 0 = 0.5 1 = 1 2 = 2 3 = 4
23…31	Reserved	0	0	x	

Notes: The Texture unit composites the fragment's Color, Texel0 and Texel1 values with one or two constant color values held in registers and passes the result on to the next unit as a texture value.
The compositing is done in two stages and is controlled separately for the RGB channels and the Alpha channel. This register defines the operation for the alpha channels in compositing stage 0 for this unit.

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

TextureCompositeFactor0

Name	Type	Offset	Format
TextureCompositeFactor0	Global *Command*	0xB328	Bitfield

Bits	Name	Read	Write	Reset	Description
0…7	red	✔	✔	x	red
8…15	green	✔	✔	x	green
16…23	blue	✔	✔	x	blue
24…31	alpha	✔	✔	x	alpha

Notes: The Texture unit composites the fragment's Color, Texel0 and Texel1 values with one or two constant color values held in registers and passes the result on to the next unit as a texture value.
The compositing is done in two stages and is controlled separately for the RGB channels and the Alpha channel. This register holds the constant factor to use with compositing stage 0.

TextureCompositeFactor1

Name	Type	Offset	Format
TextureCompositeFactor1	Texture *Command*	0xB330	Bitfield

Bits	Name	Read	Write	Reset	Description
0…7	red	✔	✔	x	red
8…15	green	✔	✔	x	green
16…23	blue	✔	✔	x	blue
24…31	alpha	✔	✔	x	alpha

Notes: The Texture unit composites the Color, Texel0 and Texel1 from a step register 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. The Delta Unit Grabs a copy of this register so the LOD source bit can be modified based on the primitive type.

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 [46]	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.

[46] 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	✔	✔	x	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	✔	✔	x	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.

TextureFormatControl
TextureFormatControlAnd
TextureFormatControlOr

Name	Type	Offset	Format
TextureFormatControl	Texture	0xD680	Bitfield
TextureFormatControlAnd	Texture	0xD688	Bitfield
TextureFormatControlOr	Texture	0xD690	Bitfield
	Control register		

Bits	Name	Read	Write	Reset	Description
0	Enable	✔	✔	x	Overall Unit enable
1	EqualInvW	✔	✔	x	1 = enable. When enabled, tests the InvW value for all vertices in the priitive. If all are equal it sets them all to 1 to improve numerical accuracy in the rasterizer.
2	WrapSA	✔	✔	x	1 = enable D3D wrapping
3	WrapTA	✔	✔	x	1 = enable D3D wrapping
4	WrapSB	✔	✔	x	1 = enable D3D wrapping
5	WrapTB	✔	✔	x	1 = enable D3D wrapping
6	Reserved	✔	✔	x	
7	ScaleByInvWA	✔	✔	x	1 = enable scaling - must be disabled in Fog unit
8	ScaleByInvWB	✔	✔	x	1 = enable scaling - must be disabled in Fog unit
9…12	Reserved	✔	✔	X	
13	PerPolyMip MapA	✔	✔	x	1 = enabled
14	PerPolyMip MapB	✔	✔	x	1 = enabled
15, 16	Reserved	✔	✔	x	
17	TextureShiftSA	✔	✔	x	1 = enabled
18	TextureShiftTA	✔	✔	x	1 = enabled
19	TextureShiftSB	✔	✔	x	1 = enabled
20	TextureShiftTB	✔	✔	x	1 = enabled
21	Enable TextureA	✔	✔	x	1 = enabled
22	Enable TextureB	✔	✔	x	1 = enable
23	EmulateMulti Texture	✔	✔	x	1 = enabled. Produces a command stream which causes the delta unit to emulate multi-texture setup.
24	Emulate3D Texture	✔	✔	x	1 = enabled. The delta unit cannot process 3D textures so a mechanism is used to process the R component first as though it is the S component, then the S and T components are processed in a second pass.

25	AALineQuad Fix	✔	✔	x	1 = when enabled, avoids a bug when drawing quad strips with polymode set to Lines using antialiasing.
26	UseDeltaMode Override	✔	✔	x	1 = enables use of the DeltaOverride* registers to reduce setup duplication during emulated multi-texturing.
27	ProjectTexLOD	✔	✔	x	1 = enabled: set LOD to 0 if texture is projected. 0 = q value is factored into the LOD calculation
28…31	Reserved	✔	✔	x	

Notes:

- Each coordinate of each texture can be individually enabled to wrap. This type of wrapping, also known as D3D wrapping, involves modifying the texture coordinates at the vertices instead of the per-pixel values after interpolation in the rasterizer. Normal texture mapping rules require that interpolation of texture co-ordinates goes from the start value to the end value in a linear fashion. If texture wrapping is enabled the rules are changed to require the interpolation to take the shortest distance between the start and end co-ordinates.
- Texture coordinates are usually divided by w during processing in the texture/fog unit. If the texture format unit does any processing (not including the multiple texture support described later) the texture coordinates must not be scaled beforehand. The scaling must be disabled in the texture/fog unit and enabled in this unit.
- Per-polyMipMap: If the rasterizer is unable to calculate the correct mipmap level for each pixel procesed, a single level of detail is approximated for the whole triangle. The value is derived from a ratio of the area the triangle takes on the scree compared to the area it takes on the texture map. The area the triangle takes on the screen can be scaled using the *TextureLODScale* parameter; there is a separate scale for each of the texture maps supported.
- Texture shift improves numerical accuracy when the rasterizer uses fixed point arithmetic for texture calculations. It is applied when texture coordinates have a large bias applied to them but still have a small range. For example, the S texture coordinates for a triangle could be 70.0, 70.3, 70.5 in which case the large integer component is redundant. This situation commonly occurs when a texture is repeated many times across a tesselated surface. When texture shift is enabled the integer component is removed, but this is only valid if the texture wrap mode (the mode used by the rasterizer) is repeat or mirror. If the texture is projected (i.e the q component is not 1.0) the shift operation is unconditionally disabled.
- EmulateMultiTexture: The emulation involves using the delta to do two passes over the data, in the first calculating texture A, in the second calculating texture B. The setup calculations for all paramters are repeated on each phase of this emulation.
- The *UseDeltaModeOverride* bit can be set which uses the registers **DeltaModeOverride0** and **DeltaModeOverride1**. Register '0' is used on the first pass (texture B or the R component of a 3D texture) and register '1' on the second pass. By setting these registers appropriately it is possible to reduce the time taken on each phase of the setup by, for example, calcuating color on the first pass and Z and specular on the second pass. The **DeltaMode** register must be restored when this optimization is disabled.
- The delta unit has a bug relating to the way anti-aliased lines are drawn when the primitive type is a quad strip. This unit works around the bug by sending a dummy register before each Render command.

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 [47]	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	Magnification Filter	✔	✔	x	This field selects the magnification filter to use. The options are 0 = Nearest 1 = Linear

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

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 [48]	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

[48] 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

TextureLodScaleA

Name	Type	Offset	Format
TextureLodScaleA	Texture *Control register*	0xD698	Float

Bits	Name	Read	Write	Reset	Description
0…31	ScaleA	✓	✓	x	IEEE Float

Notes: Scales the area the triangle takes on screen prior to deriving the LOD ratio for texture A when *PerPolyMipMap** is enabled in **TextureFormatControl**

TextureLodScaleB

Name	Type	Offset	Format
TextureLodScaleB	Texture *Control register*	0xD6A0	Float

Bits	Name	Read	Write	Reset	Description
0…31	ScaleB	✓	✓	x	IEEE Float

Notes: Scales the area the triangle takes on screen prior to deriving the LOD ratio for texture B when *PerPolyMipMap** is enabled in **TextureFormatControl**

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.

TextureMode
TextureModeAnd
TextureModeOr

Name	Type	Offset	Format
TextureMode	Texture Fog	0x9558	Bitfield
TextureModeAnd	Texture Fog	0xAB60	Bitfield
TextureModeOr	Texture Fog	0xAB68	Bitfield

Control register

Bits	Name	Read	Write	Reset	Description
0	Enable	✔	✔	x	When set will cause texture processing to be done and the Multi Texture Coordinate for this texture will be generated. The TextureEnable bit in the Begin Command must also be set enable this processing.
1	Transform Enable	✔	✔	x	When set causes the incoming texture or the texture generated from the TexGen operation to be multiplied by the Texture matrix. Frequently the texture matrix will be a unit matrix so the transformation can be preferably avoided.
2, 3	TexGenMode S	✔	✔	x	This field controls the automatic generation of texture coordinates for the S texture component from the vertex or normal information. The TexGen operations are: 0 Normal. 1 ObjectLinear. 2 EyeLinear. 3 SphereMap. These are described later.
4, 5	TexGenMode T	✔	✔	x	This field controls the automatic generation of texture coordinates for the T texture component from the vertex or normal information. The TexGen operations are: 0 Normal. 1 ObjectLinear. 2 EyeLinear. 3 SphereMap. These are described later.
6, 7	TexGenMode R	✔	✔	x	This field controls the automatic generation of texture coordinates for the R texture component from the vertex or normal information. The TexGen operations are: 0 Normal. 1 ObjectLinear. 2 EyeLinear. 3 SphereMap (illegal for OpenGL). These are described later.

8, 9	TexGenMode Q	✔	✔	x	This field controls the automatic generation of texture coordinates for the Q texture component from the vertex or normal information. The TexGen operations are: 0 Normal. 1 ObjectLinear. 2 EyeLinear. 3 None (SphereMap is illegal). These are described later.
10	TexGenS	✔	✔	x	When this bit is set the S component of the texture coordinate is generated automatically, otherwise it is taken from the current texture S value. This only has an effect when the TexGen operation is ObjectLinear, EyeLinear or SphereMap.
11	TexGenT	✔	✔	x	When this bit is set the T component of the texture coordinate is generated automatically, otherwise it is taken from the current texture T value. This only has an effect when the TexGen operation is ObjectLinear, EyeLinear or SphereMap.
12	TexGenR	✔	✔	x	When this bit is set the R component of the texture coordinate is generated automatically, otherwise it is taken from the current texture R value. This only has an effect when the TexGen operation is ObjectLinear or EyeLinear.
13	TexGenQ	✔	✔	X	When this bit is set the Q component of the texture coordinate is generated automatically, otherwise it is taken from the current texture Q value. This only has an effect when the TexGen operation is ObjectLinear or EyeLinear.
14…16	WhichCurrent TextureCoord	✔	✔	x	This field selects which current texture coordinates are to be used during texture coordinate generation.
17…19	WhichTexGen	✔	✔	x	This field selects which texture generation coefficients are to be used during any texture generation operation selected for this texture.
20…22	WhichMatrix	✔	✔	x	This field selects which matrix is to be used to transform the computed texture coordinate.
23	Feedback Enable	✔	✔	x	This bit, when set, will cause this texture coordiante to be calculated and output during feedback mode irrespective of the enable state.
24	UserInvW	✔	✔	x	This bit, when set, takes the invW value from the w field on the input vertex rather calculating the reciprocal of w itself. A typical use of this is in D3D when post transformed and clipped vertices are provided as (x, y, z, 1/w) in screen coordinates. This bit is only used from TextureMode register 0 - it is present in the other TextureMode registers and can be read and written to, but has no effect.
25	BlendEnable	✔	✔	x	This bit, when set, causes this texture and its paired texture to be blended together using the TBFactor0 and TBFactor1 values. This bit is only used from TextureMode register 0…3 - it is present in the other TextureMode registers and can be read and written to, but has no effect.

26	NoDivByW	✔	✔	x	This bit, when set, prevents the s, t, r and q values of a texture from being divided through by the homogeneous w value. This would normally be set when the Texture Format Unit is being used to emulate missing functionality in the Delta Unit. The Texture Format Unit will do the necessary divide by w. This bit is only used from TextureMode register 0 - it is present in the other TextureMode registers and can be read and written to, but has no effect.
27	D3Dsphere Map	✔	✔	x	This bit, when set, will use the D3D sphere mapping equations instead of the OpenGL equations. This bit is only used from TextureMode register 0 - it is present in the other TextureMode registers and can be read and written to, but has no effect.
28	D3Dnon LocalViewer	✔	✔	x	This bit, when set, will force the eyePosition in the reflection calculation (SphereMap) to be replaced with (0, 0, -1). This bit is only used from TextureMode register 0 - it is present in the other TextureMode registers and can be read and written to, but has no effect.
29	D3DforceQ ToOne	✔	✔	x	This bit, when set, will force the texture q coordinate after transformantion to be one.
28…31	Reserved	✔	✔	x	

Notes: The unit's texture operation is controlled by the TextureMode message. This message will update mode the register selected by the TextureModeSelect message.

TextureModeSelect

Name	**Type**	**Offset**	**Format**
TextureModeSelect	Texture Fog *Control register* Broadcast	0x9540	Integer

Bits	**Name**	**Read**	**Write**	**Reset**	**Description**
0…3	TextureMode *n*	✓	✓	x	
4…31	Reserved	0	0	x	

Notes: The 8 sets of mode, TexGen and matrices are not directly accessible but are programmed through a common set of tags. The targets are set up in the **TextureModeSelect**, **TextureTexGenSelect** and **TextureMatrixSelect** registers for mode, **TexGen** and matrix targets respectively
This register holds which of the 8 texture mode registers is to be updated on subsequent **TextureMode**, **TextureModeAnd** or **TextureModeOr** commands.

.

TextureReadMode0
TextureReadMode0And
TextureReadMode0Or

Name	Type	Offset	Format
TextureReadMode0	Texture	0xB400	Bitfield
TextureReadMode0And	Texture	0xAC30	Bitfield Logic Mask
TextureReadMode0Or	Texture	0xAC38	Bitfield Logic Mask
	Control registers		

Bits	Name	Read [49]	Write	Reset	Description
0	Enable	✔	✔	x	When set causes any texels needed by the fragment to be read. This is also qualified by the TextureEnable bit in the *Render* command.
1…4	Width	✔	✔	x	This field holds the width of the map as a power of two. The legal range of values for this field is 0 (map width = 1) to 11 (map width = 2048). This is only used when Texture3D is enabled and then is only used for cache management purposes and *not* for address calculations.
5…8	Height	✔	✔	x	This field holds the height of the map as a power of two. The legal range of values for this field is 0 (map height = 1) to 11 (map height = 2048). This is only used when Texture3D is enabled and then is only used for cache management purposes and *not* for address calculations.
9…10	TexelSize	✔	✔	x	This field holds the size of the texels in the texture map. The options are: 0 = 8 bits 1 = 16 bits 2 = 32 bits 3 = 64 bits (Only valid for spans)
11	Textue3D	✔	✔	x	This bit, when set, enables 3D texture index generation. The CombinedCache mode bit should not be set when 3D textures are being used.
12	CombineCache	✔	✔	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.

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

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.
- 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 [50]	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.

[50] 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.

TextureTexGenSelect

Name	Type	Offset	Format
TextureTexGenSelect	Texture Fog *Control register* Broadcast	0x9550	Integer

Bits	Name	Read	Write	Reset	Description
0…3	TextureGen *n*	✓	✓	x	
4…31	Reserved	0	0	x	

Notes: The 8 sets of mode, TexGen and matrices are not directly accessible but are programmed through a common set of tags. The targets are set up in the TextureModeSelect, TextureTexGenSelect and TextureMatrixSelect registers for mode, TexGen and matrix targets respectively.
This message holds which of the 8 sets of texture generation coefficients are to be updated on subsequent **TexGen[0…15]** commands.

TFq

Name	Type	Offset	Format
TFq	Matrix *Control register*	0xC4C0	Float

Bits	Name	Read	Write	Reset	Description
0..31	Qvalue	✗	✓	x	Q component of incoming Texture F

Notes: The Q component supplied to build wide vertex data - see **TFs1**

TFr

Name	Type	Offset	Format
TFr	Matrix *Control register*	0xC4C8	Float

Bits	Name	Read	Write	Reset	Description
0..31	Rvalue	✗	✓	x	R component of incoming Texture F

Notes: The R component supplied to build wide vertex data - see **TFs1**

TFs1
TFs2
TFs3
TFs3q
TFs4

Name	Type	Offset	Format
TFs1	Matrix	0xC4D8	Wide
TFs2	Matrix	0xC4E0	Wide
TFs3	Matrix	0xC4E8	Wide
TFs3q	Matrix	0xC4F8	Wide
TFs4	Matrix	0xC4F0	Wide
	Control register		

Bits	Name	Read [51]	Write	Reset	Description
0..31	S	✗	✔	x	Texture F co-ordinate S component
32..63	T	✗	✔	x	Texture F co-ordinate T component
64..95	R	✗	✔	x	Texture F co-ordinate R component
96..127	Q	✗	✔	x	Texture F co-ordinate Q component

Notes: These registers hold the incoming texture F coordinate, and the coordinate has 1 (s), 2 (s, t), 3 (s, t, r) or 4 (s, t, r, q) components. The missing t and r components are replaced with 0 and the missing q by 1. The register is used on output. **TFs3q** has r missing and not q as might be expected

TFt

Name	Type	Offset	Format
TFt	Matrix	0xC4D0	Float
	Control register		

Bits	Name	Read	Write	Reset	Description
0..31	Tvalue	✗	✓	x	T component of incoming Texture F

Notes: The T component supplied to build wide vertex data - see **TFs1**

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

TGq

Name	Type	Offset	Format
TGq	Matrix *Control register*	0xC500	Float

Bits	Name	Read	Write	Reset	Description
0..31	Qvalue	✕	✓	x	Q component of incoming Texture G

Notes: The Q component supplied to build wide vertex data - see **TGs1**

TGr

Name	Type	Offset	Format
TGr	Matrix *Control register*	0xC508	Float

Bits	Name	Read	Write	Reset	Description
0..31	Rvalue	✕	✓	x	R component of incoming Texture G

Notes: The R component supplied to build wide vertex data - see **TGs1**

TGs1
TGs2
TGs3
TGs3q
TGs4

Name	Type	Offset	Format
TGs1	Matrix	0xC518	Wide
TGs2	Matrix	0xC520	Wide
TGs3	Matrix	0xC528	Wide
TGs3q	Matrix	0xC538	Wide
TGs4	Matrix	0xC530	Wide
	Control register		

Bits	Name	Read [52]	Write	Reset	Description
0..31	S	✗	✔	x	Texture G co-ordinaTG S component
32..63	T	✗	✔	x	Texture G co-ordinaTG T component
64..95	R	✗	✔	x	Texture G co-ordinaTG R component
96..127	Q	✗	✔	x	Texture G co-ordinaTG Q component

Notes: These registers hold the incoming Texture G coordinaTG, and the coordinaTG has 1 (s), 2 (s, t), 3 (s, t, r) or 4 (s, t, r, q) components. The missing t and r components are replaced with 0 and the missing q by 1. The register is used on output. **TGs3q** has r missing and not q as might be expecTGd

TGt

Name	Type	Offset	Format
TGt	Matrix	0xC510	Float
	Control register		

Bits	Name	Read	Write	Reset	Description
0..31	Tvalue	✗	✓	x	T component of incoming Texture G

Notes: The T component supplied to build wide vertex data - see **TGs1**

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

THq

Name	Type	Offset	Format
THq	Matrix *Control register*	0xC540	Float

Bits	Name	Read	Write	Reset	Description
0..31	Qvalue	✕	✓	x	Q component of incoming Texture H

Notes: The Q component supplied to build wide vertex data - see **THs1**

THr

Name	Type	Offset	Format
THr	Matrix *Control register*	0xC548	Float

Bits	Name	Read	Write	Reset	Description
0..31	Rvalue	✕	✓	x	R component of incoming Texture H

Notes: The R component supplied to build wide vertex data - see **THs1**

THs1
THs2
THs3
THs3q
THs4

Name	Type	Offset	Format
THs1	Matrix	0xC558	Wide
THs2	Matrix	0xC560	Wide
THs3	Matrix	0xC568	Wide
THs3q	Matrix	0xC578	Wide
THs4	Matrix	0xC570	Wide
	Control register		

Bits	Name	Read [53]	Write	Reset	Description
0..31	S	✗	✔	x	Texture H co-ordinate S component
32..63	T	✗	✔	x	Texture H co-ordinate T component
64..95	R	✗	✔	x	Texture H co-ordinate R component
96..127	Q	✗	✔	x	Texture H co-ordinate Q component

Notes: These registers hold the incoming texture A coordinate, and the coordinate has 1 (s), 2 (s, t), 3 (s, t, r) or 4 (s, t, r, q) components. The missing t and r components are replaced with 0 and the missing q by 1. The register is used on output. **THs3q** has r missing and not q as might be expected

THt

Name	Type	Offset	Format
THt	Matrix	0xC550	Float
	Control register		

Bits	Name	Read	Write	Reset	Description
0..31	Tvalue	✗	✓	x	T component of incoming Texture H

Notes: The T component supplied to build wide vertex data - see **THs1**

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

TMask

Name	Type	Offset	Format
TMask	RectangleDMA *Command*	0xCED0	Bitfield

Bits	Name	Read	Write	Reset	Description
0..7	T	✔	✔	x	Bit mask specifying which R5 units are to forward messages to the next R5.
8..15	S	✔	✔	x	Bit mask specifying which R5 units are to process Sync Data
16..23	F	✔	✔	x	Bit mask specifying which R5 units are to forward messages to their own pipeline
24..31	reserved				

Notes:

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).

Tq4

Name	Type	Offset	Format
Tq4	RectangleDMA *Control register*	0x98A8	integer

Bits	Name	Read	Write	Reset	Description
0…31	Q4	✓	✓	x	

Notes: **Tq4, Tr4, Ts1, Ts2, Ts4, Tt2** and **Tt4** hold the s, t, r and q texture components. The Ts* registers must be written last because they trigger the generation of the internal wide tag for the texture. All the texture components must be written together and not interleaved with writes to color, face normal, normal or vertex registers.
Writing to **Ts1** ignores any supplied values for other components and sets them to 0.0 or (vertex) 1.0.
The t component can be written using either **Tt2** or **Tt4**, but the tags are grouped with the Ts2 and Ts4 registers so the choice would normally be consistent.
A texture is optionally transformed using the **TextureMatrix** command. See also **TransformMode**

Tr4

Name	Type	Offset	Format
Tr4	RectangleDMA *Control register*	0x98A8	integer

Bits	Name	Read	Write	Reset	Description
0…31	R4	✓	✓	x	

Notes: **Tq4, Tr4, Ts1, Ts2, Ts4, Tt2** and **Tt4** hold the s, t, r and q texture components. The Ts* registers must be written last because they trigger the generation of the internal wide tag for the texture. All the texture components must be written together and not interleaved with writes to color, face normal, normal or vertex registers.
Writing to **Ts1** ignores any supplied values for other components and sets them to 0.0 or (vertex) 1.0.
The t component can be written using either **Tt2** or **Tt4**, but the tags are grouped with the Ts2 and Ts4 registers so the choice would normally be consistent.
A texture is optionally transformed using the **TextureMatrix** command. See also **TransformMode**

TransformCurrent

Name	Type	Offset	Format
TransformCurrent	Transformation *Control register*	0x98A8	bitmask

Bits	Name	Read	Write	Reset	Description
0	Normal	✓	✓	x	Bit 0, when set, enables the transformation of the normal.
1	Face normal	✓	✓		Bit 1, when set, enables the transformation of the face normal.
2	Texture	✓	✓		Bit 2, when set, enables the transformation of the texture.
3	Color	✓	✓		Bit3, when set, enables the current colour to be refreshed.
4…31	Unused	✓	✓		

Notes: This message causes the current normal, face normal and texture coordinate to be transformed. The bottom three bits enable the individual transformations and these are further qualified by the corresponding fields in the TransformMode message.

TransformCurrent	2B	7	PICA		

TransformMode
TransformModeAnd
TransformModeOr

Name	Type	Offset	Format
TransformMode	Transform		integer
TransformModeAnd	Transform	0xAA80	Bitfield Logic Mask
TransformModeOr	Transform	0xAA88	Bitfield Logic Mask
	Control register		*Broadcast Mode*

Bits	Name	Read	Write	Reset	Description
0	UseModelView Matrix	✓	✓	x	When set causes the incoming vertex to be multiplied by the ModelView matrix. This is only necessary if the vertex in eye space is needed for subsequent processing. A small gain in performance will be seen when this transformation is disabled. The eye space vertex is used for EyeLinear TexGen, user clipping planes, fog, lighting, or auto generation of the face normal..

1	UseModel View Projection Matrix	✓	✓		When set causes the incoming vertex to be multiplied by the ModelViewProjection matrix to calculate coordinates in clip space. This bit should normally be set.
2	Transform Normal	✓	✓		When set causes any incoming vertex normal to be multiplied by the Normal matrix. The normal transformation happens in a different unit.
3	Transform FaceNormal	✓	✓		When set causes any incoming face normal to be multiplied by the Normal matrix. This only needs to be set if face normal backface test is enabled.
4…16	Reserved	✓	✕		
17	BlendVertex	✓	✓		This bit, when set, enables the blending of vertices before transformation.
18…31	Reserved	✓	✕		

Notes: 1) Transforming Vertices: The incoming vertex in the Vertex message is optionally transformed by zero, one or two matrices (4x4) under control of the *UseModelViewMatrix* and *UseModelViewProjectionMatrix* bits in the **TransformMode** register. The ModelView matrix is used to generate eye space vertex coordinates which are required by some geometry and lighting configurations. The ModelViewProjection matrix generates clip space vertex coordinates and should normally be enabled.

Note: There is no matrix stack or mechanism to concatenate matrices in this unit so this must be done by software.

2) Transforming Normals. The incoming FaceNormal is transformed by zero or one matrix (3x3) under control of the *TransformFaceNormal* bit in the **TransformMode** register. After the transformation the face normal can be optionally normalised, however this is done in a separate unit. The face normal never needs to be normalised if it is only going to be used for backface culling.

TriangleClipAreaThreshold

Name	**Type**	**Offset**	**Format**
TriangleClipAreaThreshold	Geometry *Control register*	0x9BE0	Float

Bits	**Name**	**Read**	**Write**	**Reset**	**Description**
0…31	area	✓	✓	x	

Notes: This is the area of a triangle in pixels for which geometric clipping will not be done, if enabled. The area is held as a floating point number and the required area should be twice the area required

TriangleExtend

Name	Type	Offset	Format
TriangleExtend	Cull *Control register* Broadcast	0xC5B0	Fixed

Bits	Name	Read	Write	Reset	Description
0...31	Scanlines	✓	✓	x	8.3 unsigned fixed point.

Notes: This message hold the number of scanlines to grow the y extent by when dealing with lines. It is updated whenever the *LineWidth* value in the Delta Unit is updated.

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	AntiAlias Enable	✓	✓	x	Enabled = 1
1	Antialias Quality	✓	✓	x	Defines the quality of antialiased triangles: 0 = 4x4 1 = 8x8

Notes:
- Enables triangle antialiasing, qualified by the *AntialiasEnable* field in the **Begin** register.
- The logic operator equivalents behave the same way but the new mode is AND'd or OR'd with the former mode before replacing it.

Ts1
TS2
TS4

Name	Type	Offset	Format
Ts1	Matrix	0x98F8	integer
Ts2	Matrix	0x9848	integer
Ts4	Matrix	0x98C0	integer
	Control register		

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

Notes: **Tq4, Tr4, Ts1, Ts2, Ts4, Tt2** and **Tt4** hold the s, t, r and q texture components. The Ts* registers must be written last because they trigger the generation of the internal wide tag for the texture. All the texture components must be written together and not interleaved with writes to color, face normal, normal or vertex registers.
Writing to **Ts1** ignores any supplied values for other components and sets them to 0.0 or (vertex) 1.0.
The t component can be written using either **Tt2** or **Tt4**, but the tags are grouped with the Ts2 and Ts4 registers so the choice would normally be consistent.
A texture is optionally transformed using the **TextureMatrix** command. See also **TransformMode**

TStart

Name	Type	Offset	Format
Tstart	Texture	0x83A0	Fixed point
	Control register		

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.

Tt2
Tt4

Name	Type	Offset	Format
Tt2	Matrix	0x9840	integer
Tt4	Matrix *Control register*	0x98B8	integer

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

Notes: **Tq4, Tr4, Ts1, Ts2, Ts4, Tt2** and **Tt4** hold the s, t, r and q texture components. The Ts* registers must be written last because they trigger the generation of the internal wide tag for the texture. All the texture components must be written together and not interleaved with writes to color, face normal, normal or vertex registers.
Writing to **Ts1** ignores any supplied values for other components and sets them to 0.0 or (vertex) 1.0.
The t component can be written using either **Tt2** or **Tt4**, but the tags are grouped with the Ts2 and Ts4 registers so the choice would normally be consistent.
A texture is optionally transformed using the **TextureMatrix** command. See also **TransformMode**

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.

UserClip0X…UserClip0W
UserClip1X…UserClip1W
UserClip2X…UserClip2W
UserClip3X…UserClip3W
UserClip4X…UserClip4W
UserClip5X…UserClip5W

Name	Type	Offset	Format
UserClip0X…UserClip0W	Geometry	0x9C08..0x9C00	float
UserClip1X…UserClip1W	Geometry	0x9C28..0x9C20	float
UserClip2X…UserClip2W	Geometry	0x9C50..0x9C48	float
UserClip3X…UserClip3W	Geometry	0x9C70..0x9C68	float
UserClip4X…UserClip4W	Geometry	0x9C90..0x9C88	float
UserClip5X…UserClip5W	Geometry	0x9CB0..0x9CA8	float
	Control Broadcast		

Bits	Name	Read	Write	Reset	Description
0…31	Reserved	✓	✓	x	.

Notes: These registers hold the plane equation coefficients for the 6 user clipping planes. The planes are enabled by the *UserClipMask* bits in the **GeometryMode** register as follows:

- There is one bit per user-defined clipping plane.
- Clipping against a plane is enabled when the corresponding bit is set.
- The clipping plane is defined in eye space by UserClipn{X | Y | Z | W}.
- Bit 0 (i.e. bit 22 in register) corresponds to UserClip0.

Vertex0

Name	Type	Offset	Format
Vertex0	Input *Control register*	0xB7B8	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Index	✔	✓	x	Index into Vertex buffer

Notes: The vertex data can be loaded without using one of the primitive types using the Vertex0, Vertex1, and Vertex2 registers. These registers specify the vertex store to load, and the data field holds the index into the array.

Vertex1

Name	Type	Offset	Format
Vertex1	Input *Control register*	0xB7C0	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Vertex	✔	✓	x	Index into Vertex buffer

Notes: The vertex data can be loaded without using one of the primitive types using the Vertex0, Vertex1, and Vertex2 registers. These registers specify the vertex store to load, and the data field holds the index into the array.

Vertex2

Name	Type	Offset	Format
Vertex2	Input *Control register*	0xB7C8	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Index	✔	✓	x	Index into Vertex buffer

Notes: The vertex data can be loaded without using one of the primitive types using the Vertex0, Vertex1, and Vertex2 registers. These registers specify the vertex store to load, and the data field holds the index into the array.

VertexBaseAddress

Name	Type	Offset	Format
VertexBaseAddress	Input *Control register*	0xB708	Integer

Bits	Name	Read	Write	Reset	Description
0…1	Reserved	0	0	x	
2…31	Address	✓	✓	x	32 bit address of base of buffer

Notes:

VertexControl

Name	Type	Offset	Format
VertexControl	Input *Control register*	0xB798	Bitfield

Bits	Name	Read	Write	Reset	Description
0-4	Size	✔	✔	x	Size of vertex in 32 words
5	CacheEnable	✔	✔	x	0 = cache off, 1 = cache on
6	Flat	✔	✔	x	0 = off, 1 = on
7	ReadAll	✔	✔	x	0 = off, 1 = on
8	SkipFlags	✔	✔	x	0 = off, 1 = on
9	OGL	✔	✔	x	0 = D3D, 1 = OGL
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:

VertexMachineMode

Name	Type	Offset	Format
VertexMachineMode	Vertex Machine *Control register*	0x9500	Bitfield

Bits	Name	Read	Write	Reset	Description
0	ObjectTagEnable	✓	✓	x	When set causes the ObjectID to be incremented by the Begin message and the result sent to the rasteriser chip.
1	ObjectIDPerPrimitive	✓	✓	x	When set causes the ObjectID to be incremented and sent to the rasterisation chip on each point, line, triangle or quad when the primitive type is set to Points, Lines, Triangles or Quads. ObjectTagEnable must be set for this to happen.
2	D3DProvokingVertex	✓	✓	x	This bit, when set, selects the D3D provoking vertex rules instead of the OpenGL rules. a triangle fan in which case use the second vertex [of each triangle].
3..31	Reserved	✗	✓	x	

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: Used for multi-pass algorithms such as emulation of multiple textures. The vertex structure holds several sets of texture coordinates for the same x,y,z coordinates and on each pass a different set is enabled by defining it as valid

ViewPortOffsetX

Name	Type	Offset	Format
ViewPort OffsetX	Geometry *Control register* Broadcast	0x9B98	Integer

Bits	Name	Read	Write	Reset	Description
0…31	offset	✔	✓	x	

Notes: The viewport mapping takes the 4D homogeneous vertex coordinates and converts them into a 3D coordinate by dividing through by the homogenous component 'w'. The resulting coordinate is often called the normalised device coordinate (or NDC)[54].
The NDC is converted to device coordinates by multiplying by the **ViewportScale** vector and then adding the **ViewportOffset** vector. The X and Y values are scaled so the true NDC range (±1.0) fills the window (or viewport) while the Z value is nominally scaled to be in the range 0.0…1.0, although the OpenGL *glDepthRange* function can be used to change this.

[54] This is a slight misnomer in this case because this is prior to clipping so the NDC may extend beyond the ±1.0 range usually associated with normalised numbers.

ViewPortOffsetY

Name	Type	Offset	Format
ViewPortOffsetY	Geometry *Control register* Broadcast	0x9BA0	Integer

Bits	Name	Read	Write	Reset	Description
0…31	offset	✔	✓	x	

Notes: The viewport mapping takes the 4D homogeneous vertex coordinates and converts them into a 3D coordinate by dividing through by the homogenous component 'w'. This is converted to device coordinates by multiplying by the **ViewportScale** vector and then adding the **ViewportOffset** vector. The X and Y values are scaled so the true NDC range (±1.0) fills the window (or viewport) while the Z value is nominally scaled to be in the range 0.0…1.0, although the OpenGL *glDepthRange* function can be used to change this.

ViewPortOffsetZ

Name	Type	Offset	Format
ViewPortOffsetZ	Geometry *Control register* Broadcast	0x9BA8	Integer

Bits	Name	Read	Write	Reset	Description
0…31	offset	✔	✓	x	

Notes: The viewport mapping takes the 4D homogeneous vertex coordinates and converts them into a 3D coordinate by dividing through by the homogenous component 'w'. This is converted to device coordinates by multiplying by the **ViewportScale** vector and then adding the **ViewportOffset** vector. The X and Y values are scaled so the true NDC range (±1.0) fills the window (or viewport) while the Z value is nominally scaled to be in the range 0.0…1.0, although the OpenGL *glDepthRange* function can be used to change this.

ViewPortScaleX

Name	Type	Offset	Format
ViewPortScaleX	Geometry *Control register* Broadcast	0x9B80	Integer

Bits	Name	Read	Write	Reset	Description
0…31	Scale	✔	✓	x	

Notes: The viewport mapping takes the 4D homogeneous vertex coordinates and converts them into a 3D coordinate by dividing through by the homogenous component 'w'. This is converted to device coordinates by multiplying by the **ViewportScale** vector and then adding the **ViewportOffset** vector. The X and Y values are scaled so the true NDC range (±1.0) fills the window (or viewport) while the Z value is nominally scaled to be in the range 0.0…1.0, although the OpenGL *glDepthRange* function can be used to change this.

ViewPortScaleY

Name	Type	Offset	Format
ViewPortScaleY	Geometry *Control register* Broadcast	0x9B88	Integer

Bits	Name	Read	Write	Reset	Description
0…31	offset	✔	✓	x	

Notes: The viewport mapping takes the 4D homogeneous vertex coordinates and converts them into a 3D coordinate by dividing through by the homogenous component 'w'. This is converted to device coordinates by multiplying by the **ViewportScale** vector and then adding the **ViewportOffset** vector. The X and Y values are scaled so the true NDC range (±1.0) fills the window (or viewport) while the Z value is nominally scaled to be in the range 0.0…1.0, although the OpenGL *glDepthRange* function can be used to change this.

ViewPortScaleZ

Name	Type	Offset	Format
ViewPortScaleZ	Geometry *Control register* Broadcast	0x9B90	Integer

Bits	Name	Read	Write	Reset	Description
0…31	offset	✔	✓	x	

Notes: The viewport mapping takes the 4D homogeneous vertex coordinates and converts them into a 3D coordinate by dividing through by the homogenous component 'w'. This is converted to device coordinates by multiplying by the **ViewportScale** vector and then adding the **ViewportOffset** vector. The X and Y values are scaled so the true NDC range (±1.0) fills the window (or viewport) while the Z value is nominally scaled to be in the range 0.0…1.0, although the OpenGL *glDepthRange* function can be used to change this.

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.

Vw

Name	Type	Offset	Format
Vw	Matrix *Control register*	0x	Bitfield

Bits	Name	Read	Write	Reset	Description
0..31	Wvalue	✓	✓	x	W component of vertex

Notes: The W vertex component supplied to build wide vertex data - see **Vx4**

Vx2

Name	Type	Offset	Format
Vx2	Command *Control register*	0x	Wide Trigger

Bits	Name	Read	Write	Reset	Description
0..31	X word	✓	✓	x	Bounding vertex X coordinate
32..64	Y word	✓	✓	x	Bounding vertex Y coordinate

Notes: The **Vy**, **Vz** and **Vw** registers hold the floating point value of the y, z and w components of the bounding vertex used to build up the bounding volume for display list viewing frustum culling. The x component is supplied in one of the **Vx2, Vx3** or **Vx4** registers. The numerical postfix defines the which of the **Bvy**, **Vz** and **Vw** registers should be used or have their default values used instead. **Vx2** generates the (x, y, 0.0, 1.0) vertex, **Vx3** generates the (x, y, z, 1.0) vertex and **Vx4** generates the (z, y, z, w) vertex. The **Vx**? register must be loaded last.

Vx3

Name	Type	Offset	Format
Vx3	Command *Control register*	0x	

Bits	Name	Read	Write	Reset	Description
0..31	X	✓	✓	x	Bounding vertex X coordinate
32..63	Y	✓	✓	x	Bounding vertex Y coordinate
64..95	Z	✓	✓	x	Bounding vertex Z coordinate

Notes: The **Vy**, **Vz** and **Vw** registers hold the floating point value of the y, z and w components of the bounding vertex used to build up the bounding volume for display list viewing frustum culling. The x component is supplied in one of the **Vx2, Vx3** or **Vx4** registers. The numerical postfix defines the which of the **Bvy**, **Vz** and **Vw** registers should be used or have their default values used instead, e.g. **Vx3** generates the (x, y, z, 1.0) vertex and **Vx4** generates the (z, y, z, w) vertex. The **Vx**? register must be loaded last.

Vx4

Name	Type	Offset	Format
Vx4	Command *Control register*	0x	

Bits	Name	Read	Write	Reset	Description
0..31	X	✗	✓	x	Bounding vertex X coordinate
32..63	Y	✗	✓	x	Bounding vertex Y coordinate
64..95	Z	✗	✓	x	Bounding vertex Z coordinate
96..127	w	✗	✓	x	Bounding vertex W coordinate

Notes: The **Vy**, **Vz** and **Vw** registers hold the floating point value of the y, z and w components of the bounding vertex used to build up the bounding volume for display list viewing frustum culling. The x component is supplied in one of the **Vx2, Vx3** or **Vx4** registers. The numerical postfix defines the which of the **Bvy**, **Vz** and **Vw** registers should be used or have their default values used instead, e.g. **Vx2** generates the (x, y, 0.0, 1.0) vertex, **Vx4** generates the (z, y, z, w) vertex. The **Vx**? register must be loaded last.

Vy

Name	Type	Offset	Format
Vy	Command *Control register*	0x	Float

Bits	Name	Read	Write	Reset	Description
0..31	Yvalue	✓	✓	x	Z component of vertex

Notes: The Y vertex component supplied to build wide vertex data - see **Vx4**

Vz

Name	Type	Offset	Format
Vz	Matrix *Control register*	0x	Float

Bits	Name	Read	Write	Reset	Description
0..31	Zvalue	✓	✓	x	Z component of vertex

Notes: The Z vertex component supplied to build wide vertex data - see **Vx4**

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	0xAB88	Bitfield
	Control register		

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 offset-sorted Region 0 Core Graphics register listings. Not all legacy registers are included.

6.1 Registers Sorted by Offset

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
DMAProfile					Rectangle DMA		Integer	×
ResetStatus					Control Status	0000	integer	×
StartXDom	0	0			Rasterizer	8000	fixed	×
dXDom	0	1	✔	×	Rasterizer	8008	fixed	×
StartXSub	0	2		×	Rasterizer	8010	fixed	×
dXSub	0	3	✔	×	Rasterizer	8018	fixed	×
StartY	0	4	×	×	Rasterizer	8020	fixed	×
dY	0	5	✔	×	Rasterizer	8028	fixed	×
Count	0	6	✔	×	Rasterizer	8030	Integer	×
Render	0	7	×	✔	Rasterizer	8038	Bitfield	✔
ContinueNewLine	0	8	×	✔	Rasterizer	8040	Integer	✔
ContinueNewDom	0	9	×	✔	Rasterizer	8048	Integer	✔
ContinueNewSub	0	A	×	✔	Rasterizer	8050	Integer	✔
Continue	0	B	×	✔	Rasterizer	8058	Integer	✔
FlushSpan	0	C	×	✔	Rasterizer	8060	tag	✔
BitMaskPattern	0	D	×	✔	Rasterizer	8068	Integer	✔×
PointTable[0…3]	1	0	✔	✔	Rasterizer	8080	bitfield	×
RasterizerMode	1	4	✔	✔	Rasterizer	80A0	Bitfield	×
YLimits	1	5	✔	✔	Rasterizer	80A8	Bitfield	×
ScanLineOwnership	1	6	✔	✔	Rasterizer	80B0		×
WaitForCompletion	1	7	×	✔	Rasterizer	80B8	Bitfield	✔
PixelSize	1	8	✔	✔	Rasterizer	80C0	Bitfield	✔
StripeOffsetY	1	9	✔	✔	Rasterizer	80C8	fixed	×
SetDeltaPort	1	E	×		Delta	80F0	bitfield	✔
ReadMonitorMode	1	F	✔	✔	Delta	80F8	bitfield	
ScissorMode	3	0	✔	✔	Scissor	8180	Bitfield	×
ScissorMinXY	3	1	✔	✔	Scissor	8188	Bitfield	×
ScissorMaxXY	3	2	✔	✔	Scissor	8190	Bitfield	×
ScreenSize	3	3	✔	✔	Scissor	8198	Bitfield	×
AreaStippleMode	3	4	✔	✔	Stipple	81A0	Bitfield	×
LineStippleMode	3	5	✔	✔	Stipple	81A8	Bitfield	
LoadLineStipple Counters	3	6	✔	✔	Stipple	81B0	Bitfield	✔
UpdateLineStippleCounters	3	7	×	✔	Stipple	81B8	Bitfield	✔

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
SaveLineStippleCounters	3	8	×	✔	Stipple	81C0	tag	✔
WindowOrigin	3	9	✔	✔	Scissor	81C8	Bitfield	×
AreaStipplePattern [0…15]	4	0	✔	✔	Stipple	8200	Bitfield	×
AreaStipplePattern [16…31]	5	0	✔	✔	Stipple	8280	Bitfield	×
FillFBWriteBufferAddr0	6	0	×	✔	2D Set Up	8300	integer	×
FillFBSourceReadBufferAddr	6	1	×	✔	2D Set Up	8308	integer	×
FillFBDestReadBufferAddr0	6	2	×	✔	2D Set Up	8310	integer	×
FillScissorMinXY	6	3	×	✔	2D Set Up	8318	fixed	×
FillScissorMaxXY	6	4	×	✔	2D Set Up	8320	fixed	×
FillForegroundColor0	6	5	×	✔	2D Set Up	8328	integer	×
FillBackgroundColor	6	6	×	✔	2D Set Up	8330	integer	×
FillConfig2D0	6	7	×	✔	2D Set Up	8338	bitfield	×
FillFBSourceReadBufferOffset	6	8	×	✔	2D Set Up	8340	integer	×
FillRectanglePosition	6	9	×	✔	2D Set Up	8348	integer	×
FillRender2D	6	A	×	✔	2D Set Up	8350	bitfield	×
FillForegroundColor1	6	B	×	✔	2D Set Up	8358	integer	×
FillConfig2D1	6	C	×	✔	2D Set Up	8360	bitfield	
FillGlyphPosition	6	D	×	✔	2D Set Up	8368	integer	×
TextureCoordMode	7	0	✔	✔	Texture coord	8380	bitfield	×
SStart	7	1	✔	✔	Texture Coord	8388	fixed	×
dSdx	7	2	✔	✔	Texture coord	8390	fixed	×
dSdyDom	7	3	✔	✔	Texture coord	8398	fixed	×
TStart	7	4	✔	✔	Texture coord	83A0	fixed	×
dTdx	7	5	✔	✔	Texture coord	83A8	fixed	×
dTdyDom	7	6	✔	✔	Texture coord	83B0	fixed	×

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
QStart	7	7	✔	✔	Texture Coord	83B8	fixed	×
dQdx	7	8	✔	✔	Texture coord	83C0	fixed	×
dQdyDom	7	9	✔	✔	Texture coord	83C8		×
LOD	7	A	✔	✔	Texture Index	83D0	fixed	×
dSdy	7	B	✔	✔	Texture coord	83D8	fixed	×
dTdy	7	C	✔	✔	Texture coord	83E0	fixed	×
dQdy	7	D	✔	✔	Texture coord	83E8	fixed	×
S1Start	8	0	✔	✔	Texture Coord	8400	fixed	×
dS1dx	8	1	✔	✔	Texture coord	8408	fixed	×
dS1dyDom	8	2	✔	✔	Texture coord	8410	fixed	×
T1Start	8	3	✔	✔	Texture coord	8418	fixed	×
dT1dx	8	4	✔	✔	Texture coord	8420	fixed	×
dT1dyDom	8	5	✔	✔	Texture coord	8428	fixed	×
Q1Start	8	6	✔	✔	Texture Coord	8430	fixed	×
dQ1dx	8	7	✔	✔	Texture coord	8438	fixed	×
dQ1dyDom	8	8	✔	✔	Texture coord	8440	fixed	×
LOD1	8	9	✔	✔	Texture Index	8448	fixed	×
TextureLodBiasS	8	A	✔	✔	Texture Index	8450	fixed	×
TextureLodBiasT	8	B	✔	✔	Texture Index	8458	fixed	×
BorderColor0	9	5	✔	✔	Texture filter	84A8	bitfield	×
LUTIndex	9	8	✔	✔	LUT	84C0	integer	×
LUTData	9	9	✔	✔	LUT	84C8	integer	×
LUTAddress	9	A	✔	✔	Texture Read	84D0	integer	×
LUTTransfer	9	B	×	✔	Texture Read	84D8	bitfield	×
TextureFilterMode	9	C	✔	✔	Texture	84E0	bitfield	×
TextureChromaUpper0	9	D	✔	✔	Color DDA	84E8	bitfield	×
TextureChromaLower0	9	E	✔	✔	Color DDA	84F0	bitfield	×
BorderColor1	9	F	✔	✔	Texture filter	84F8	bitfield	×

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
TextureBaseAddr[16]	0A	0	✔	✔	Texture Read	8500	integer	×
TextureMapWidth[16]	0B	0	✔	✔	Texture Read	8580	bitfield	×
TextureChromaUpper1	0C	0	✔	✔	Texture Filter	8600	bitfield	×
TextureChromaLower1	0C	1	✔	✔	Texture Filter	8608	bitfield	×
TextureApplicationMode	0D	0	✔	✔	Texture Application	8680	bitfield	×
TextureEnvColor	0D	1	✔	✔	Texture	8688	bitfield	×
FogMode	0D	2	✔	✔	Fog	8690	bitfield	×
FogColor	0D	3	✔	✔	Fog	8698	fixed	×
Fstart	0D	4	✔	✔	Fog	86A0	fixed	×
DFdx	0D	5	✔	✔	Fog	86A8	fixed	×
DFdyDom	0D	6	✔	✔	Fog	86B0	fixed	×
ZFogBias	0D	7	✔	✔	Fog	86B8	float	×
TextTGlyphAddr0	0E	0	×	✔	Rasterizer	8700	integer	×
TextRender2DGlyph0	0E	1	×	✔	Rasterizer	8708	bitfield	✔
TextTGlyphAddr1	0E	2	×	✔	Rasterizer	8710	integer	×
TextRender2DGlyph1	0E	3	×	✔	Rasterizer	8718	bitfield	✔
TextTGlyphAddr2	0E	4	×	✔	Rasterizer	8720	integer	×
TextRender2DGlyph2	0E	5	×	✔	Rasterizer	8728	bitfield	✔
TextTGlyphAddr3	0E	6	×	✔	Rasterizer	8730	integer	×
TextRender2DGlyph3	0E	7	×	✔	Rasterizer	8738	bitfield	✔
TextTGlyphAddr4	0E	8	×	✔	Rasterizer	8740	integer	×
TextRender2DGlyph4	0E	9	×	✔	Rasterizer	8748	bitfield	✔
TextTGlyphAddr5	0E	A	×	✔	Rasterizer	8750	integer	×
TextRender2DGlyph5	0E	B	×	✔	Rasterizer	8758	bitfield	✔
TextTGlyphAddr6	0E	C	×	✔	Rasterizer	8760	integer	×
TextRender2DGlyph6	0E	D	×	✔	Rasterizer	8768	bitfield	✔

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
TextTGlyphAddr7	0E	E	×	✔	Rasterizer	8770	integer	×
TextRender2DGlyph7	0E	F	×	✔	Rasterizer	8778	bitfield	✔
RStart	0F	0	✔	✔	Color DDA	8780	fixed	×
dRdx	0F	1	✔	✔	Color DDA	8788	fixed	×
dRdyDom	0F	2	✔	✔	Color DDA Delta	8790	fixed	×
GStart	0F	3	✔	✔	Color DDA	8798	fixed	×
DGdx	0F	4	✔	✔	Color DDA	87A0	fixed	×
DGdyDom	0F	5	✔	✔	Color DDA	87A8	fixed	×
BStart	0F	6	✔	✔	Color DDA	87B0	fixed	×
dBdx	0F	7	✔	✔	Color DDA	87B8	fixed	×
dBdyDom	0F	8	✔	✔	Color DDA	87C0	fixed	×
AStart	0F	9	✔	✔	Color DDA	87C8	fixed	×
DAdx	0F	A	✔	✔	Color DDA	87D0	fixed	×
DAdyDom	0F	B	✔	✔	Color DDA	87D8	fixed	×
ColorDDAMode	0F	C	✔	✔	Color DDA	87E0	bitfield	
ConstantColor	0F	D	✔	✔	Color DDA	87E8	bitfield	
Color	0F	E	×	✔	Color DDA	87F0		✔×
AlphaTestMode	10	0	✔	✔	Alpha Blend & Alpha Test	8800	bitfield	×
AntialiasMode	10	1	✔	✔	Alpha test	8808	bitfield	×
DitherMode	10	3	✔	✔	Dither	8818	bitfield	×
FBSoftwareWriteMask	10	4	✔	✔	Logic Ops	8820	integer	×
LogicalOpMode	10	5	✔	✔	Logic Op	8828	bitfield	×
RouterMode	10	8	✔	✔	Router	8840	bitfield	×
LBReadFormat	11	1	✔	✔	LB Read	8888	bitfield	×
LBStencil	11	5	×	✔	Host Out	88A8	bitfield	×
LBDepth	11	6	×	✔	LB Read/Host Out	88B0	integer	×
LBWriteMode	11	8	✔	✔	LB Write	88C0	bitfield	×
LBWriteFormat	11	9	✔	✔	LB Write	88C8	bitfield	×
TextureData	11	D	×	✔	Localbuffer R/W	88E8	integer	×

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
HostInID	12	0	✔	✔	Host In	8900		×
Security	12	1	×	✔	Host In	8908	bitfield	×
FlushWriteCombining	12	2	×	✔	Host In	8910	integer	×
HostInState	12	3	✔	✔	Host In	8918	integer	×
HostInDMAAddress	12	7	✔	✔	Host In	8938	integer	×
HostInState2	12	8	✔	✔	Host In	8940	integer	×
Window	13	0	✔	✔	Stencil	8980	bitfield	×
StencilMode	13	1	✔	✔	Stencil	8988	bitfield	×
StencilData	13	2	✔	✔	Stencil	8990	bitfield	×
Stencil	13	3	✔	✔	Stencil	8998	bitfield	✔
DepthMode	13	4	✔	✔	Depth	89A0	bitfield	×
Depth	13	5	✔	✔	Depth	89A8	integer	✔×
ZStartU	13	6	✔	✔	Depth	89B0	fixed	×
ZStartL	13	7	✔	✔	Depth & Fog	89B8	fixed	×
dZdxU	13	8	✔	✔	Depth & Fog	89C0	fixed	×
dZdxL	13	9	✔	✔	Depth & Fog	89C8	fixed	×
dZdyDomU	13	A	✔	✔	Depth & Fog	89D0	fixed	×
dZdyDomL	13	B	✔	✔	Depth & Fog	89D8	bitfield	×
FBColor	15	3	0	×	FB Write	8A98	n/a	×
FBWriteMode	15	7	✔	✔	FB Write	8AB8	bitfield	×
FBHardwareWriteMask	15	8	✔	✔	FB Write	8AC0	mask	×
FBBlockColor	15	9	✔	✔	FB Read/Write & Logic Ops	8AC8	integer	×
FilterMode	18	0	✔	✔	Host Out	8C00	bitfield	×
StatisticMode	18	1	✔	✔	Host Out	8C08	bitfield	×
MinRegion	18	2	✔	✔	Host Out	8C10	bitfield	×
MaxRegion	18	3	✔	✔	Host Out	8C18	bitfield	
ResetPickResult	18	4	×	✔	Host Out	8C20	tag	✔
MinHitRegion	18	5	×	✔	Host Out	8C28	bitfield	✔
MaxHitRegion	18	6	×	✔	Host Out	8C30	bitfield	✔
PickResult	18	7	×	✔	Host Out	8C38	bitfield	✔
Sync	18	8	×	✔	Host Out	8C40	bitfield	✔
RLEMask	18	9	✔	✔	Host Out	8C48	bitfield	×

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
SuspendUntilFrameBlank	18	F	×	✔	Framebuffer Write	8C78	bitfield	✔
KsRStart	19	0	✔	✔	Texture Application	8C80	fixed	×
dKsRdx	19	1	✔	✔	Texture	8C88	fixed	×
dKsRdyDom	19	2	✔	✔	Texture	8C90	fixed	×
KsGStart	19	3	✔	✔	Texture Application	8C98	fixed	×
dKsGdx	19	4	✔	✔	Texture	8CA0	fixed	×
dKsGdyDom	19	5	✔	✔	Texture	8CA8	fixed	×
KsBStart	19	6	✔	✔	Texture Application	8CB0	fixed	×
dKsBdx	19	7	✔	✔	Texture	8CB8	fixed	×
dKsBdyDom	19	8	✔	✔	Texture	8CC0	fixed	×
KdRStart	1A	0	✔	✔	Texture	8D00	fixed	×
dKdRdx	1A	1	✔	✔	Texture	8D08	fixed	×
dKdRdyDom	1A	2	✔	✔	Texture	8D10	fixed	×
KdGStart	1A	3	✔	✔	Texture	8D18	fixed	×
DKdGdx	1A	4	✔	✔	Texture	8D20	fixed	×
dKdGdyDom	1A	5	✔	✔	Texture	8D28	fixed	×
KdBStart	1A	6	✔	✔	Texture	8D30	fixed	×
DKdBdx	1A	7	✔	✔	Texture	8D38	fixed	×
DKdBdyDom	1A	8	✔	✔	Texture	8D40	fixed	×
ContextDump	1B	8	×	✔	Global	8DC0	bitfield	✔
ContextRestore	1B	9	×	✔	Global	8DC8	bitfield	✔
ContextData	1B	A	×	✔	Global	8DD0	bitfield	×
LUT[0…15]	1D	0	✔	✔	LUT	8E80	bitfield	×
YUVMode	1E	0	✔	✔	YUV Unit	8F00	bitfield	×
ChromaUpper	1E	1	✔	✔	Color DDA & Alpha Blend	8F08	bitfield	×
ChromaLower	1E	2	✔	✔	Color DDA & Alpha Blend	8F10	bitfield	×
ChromaTestMode	1E	3	✔	✔	Color DDA & Alpha Blend	8F18	bitfield	×
FeedbackToken	1F	0	×	✔	Geometry	8F80		×
FeedbackX	1F	1	×	✔	Geometry	8F88	integer	×
FeedbackY	1F	2	×	✔	Geometry	8F90	integer	×

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
FeedbackZ	1F	3	×	✔	Geometry	8F98	integer	×
FeedbackW	1F	4	×	✔	Geometry	8FA0	integer	×
FeedbackRed	1F	5	×	✔	Material	8FA8	integer	×
FeedbackGreen	1F	6	×	✔	Material	8FB0	integer	×
FeedbackBlue	1F	7	×	✔	Material	8FB8	integer	×
FeedbackAlpha	1F	8	×	✔	Material	8FC0	integer	×
FeedbackS	1F	9	×	✔	Texture/fog	8FC8	integer	×
FeedbackT	1F	A	×	✔	Texture/fog	8FD0	integer	×
FeedbackR	1F	B	×	✔	Texture/fog	8FD8	integer	×
FeedbackQ	1F	C	×	✔	Texture/fog	8FE0	integer	×
SelectRecord	1F	D		✔	Geometry	8FE8	integer	×
PassThrough	1F	E		✔	Host Out	8FF0	integer	×
EndOfFeedback	1F	F	×	✔	Host Out	8FF8		×
DeltaMode	26	0	✔	✔	Delta	9300	bitfield	×
DrawTriangle	26	1	×	✔	Delta	9308	bitfield	✔
RepeatTriangle	26	2	×	✔	Delta	9310	tag	✔
DrawLine01	26	3	×	✔	Delta	9318	fixed	✔
DrawLine10	26	4	×	✔	Delta	9320	fixed	✔
RepeatLine	26	5	×	✔	Delta	9328	tag	✔
YBias	29	1	✔	✔	Delta	9488	float	×
PointMode	29	2	✔	✔	Delta	9490		×
PointSize	29	3	✔	✔	Delta	9498	int	×
AAPointSize	29	4	✔	✔	Delta	94A0	float	×
LineMode	29	5	✔	✔	Delta	94A8	bitfield	×
LineWidth	29	6	✔	✔	Delta	94B0	int	×
LineWidthOffset	29	7	✔	✔	Delta	94B8	int	×
AALineWidth	29	8	✔	✔	Delta	94C0	float	×
TriangleMode	29	9	✔	✔	Delta	94C8	float	×
RectangleMode	29	A	✔	✔	Delta	94D0		×
RectangleWidth	29	B	✔	✔	Delta	94D8	float	×
RectangleHeight	29	C	✔	✔	Delta	94E0	float	×
Rectangle2DMode	29	D	✔	✔	Delta	94E8	bitfield	×
Rectangle2DControl	29	E	✔	✔	Delta	94F0	bitfield	×
VertexMachineMode	2A	0	✔	✔	Vertex Machine	9500		B
GeometryMode	2A	2	✔	✔	Geometry	9510	bitfield	B
NormaliseMode	2A	3	✔	✔	Normalisation	9518		B

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
LightingMode	2A	4	✔	✔	Lighting	9520		B
ColorMaterialMode	2A	5	✔	✔	Material	9528		B
MaterialMode	2A	6	✔	✔	Material	9530		B
FogVertexMode	2A	7	✔	✔	Tx/Fog	9538		B
TextureModeSelect	2A	8	✔	✔		9540		B
TextureTexGenSelect	2A	A	✔	✔	Texture Fog	9550	int	B
TextureMode	2A	B	✔	✔		9558		B
PipeMode	2A	C	✔	✔	Pipe Manager	9560		B
StripeFilterMode	2A	D	✔	✔	Cull	9568		B
MatrixMode	2A	E	✔	✔	Matrix	9570		B
GammaFilter	2A	F	✔	✔	Matrix	9578		B
SelectResult	2B	0	✔	×	Geometry	9580		×
Begin	2B	2	×	✔	Delta	9590	bitfield	×
End	2B	3	×	✔	Delta	9598	birfield	×
EdgeFlag	2B	4	✔	✔	Vertex machine	95A0		
ObjectIDValue	2B	5	✔	✔		95A8		
IncrementObjectID	2B	6	✔	✔	PipeManager	95B0		✔
SaveCurrent	2B	9	✔	✔		95C8		
RestoreCurrent	2B	A	✔	✔		95D0		
InitNames	2B	B	✔	✔	Geometry	95D8		
PushName	2B	C	✔	✔		95E0		
PopName	2B	D	✔	✔		95E8		
LoadName	2B	E	✔	✔	Geometry	95F0		
GeomRectangle	2D	4	×	✔	Geometry	96A0		✓
SetLightPipe	2D	F	×	✔	Light Mux	96F8		
DrawRectangle2D	2F	4	×		Delta	97A0	bitfield	✔
Nz	30	0	✔	✔	Pipe Manager	9800		×
Ny	30	1	✔	✔	Pipe Manager	9808		×
Nx	30	2	✔	✔	Pipe Manager	9810		×
Ca	30	3	✔	✔	Command	9818		×
Cb	30	4	✔	✔	Command	9820		×
Cg	30	5	✔	✔	Command	9828		×

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
Cr3	30	6	✔	✔	Pipe Manager	9830		×
Cr4	30	7	✔	✔	Pipe Manager	9838		×
Tt2	30	8	✔	✔	Texture Fog	9840		×
Ts2	30	9	✔	✔	Texture Fog	9848		×
Vw	30	A	✔	✔	Pipe Manager	9850		×
Vz	30	B	✔	✔	Pipe Manager	9858		×
Vy	30	C	✔	✔	Pipe Manager	9860		×
Vx2	30	D	✔	✔	Pipe Manager	9868		×
Vx3	30	E	✔	✔	Pipe Manager	9870		×
Vx4	30	F	✔	✔	Pipe Manager	9878		×
FNz	31	0	✔	✔	Matrix	9880		×
Fny	31	1	✔	✔	Matrix	9888		×
FNx	31	2	✔	✔	Matrix	9890		×
PackedColor3	31	3	✔	✔	Pipe Manager	9898	int	×
PackedColor4	31	4	✔	✔	Pipe Manager	98A0	int	×
Tq4	31	5	✔	✔	Texture Fog	98A8		×
Tr4	31	6	✔	✔	Texture Fog	98B0		×
Tt4	31	7	✔	✔	Texture Fog	98B8		×
Ts4	31	8	✔	✔	Texture Fog	98C0		×
RPw	31	9	✔	✔	Matrix	98C8		×
RPz	31	A	✔	✔	Matrix	98D0		×
Rpy	31	B	✔	✔	Matrix	98D8		×
RPx2	31	C	✔	✔	Matrix	98E0		×
RPx3	31	D	✔	✔	Matrix	98E8		×
RPx4	31	E	✔	✔	Matrix	98F0		×
Ts1	31	F	✔	✔	Texture Fog	98F8		×

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
ModelViewMatrix[16]	32	0	✔	✔	Pipe Manager	9900		B
ModelViewProjectionMatrix[16]	33	0	✔	✔	Pipe Manager	9980		B
NormalMatrix[9]	34	0	✔	✔	Transformation Unit	9A00		B
TexGen[16]	36	0	✔	✔	Texture Fog	9B00		B
ViewPortScaleX	37	0	✔	✔	Geometry	9B80	float	B
ViewPortScaleY	37	1	✔	✔	Geometry	9B88	float	B
ViewPortScaleZ	37	2	✔	✔	Geometry	9B90	float	B
ViewPortOffsetX	37	3	✔	✔	Geometry	9B98	float	B
ViewPortOffsetY	37	4	✔	✔	Geometry	9BA0	float	B
ViewPortOffsetZ	37	5	✔	✔	Geometry	9BA8	float	B
FogDensity	37	6	✔	✔	Tx/Fog	9BB0	float	B
FogScale	37	7	✔	✔	Tx/Fog	9BB8		B
FogEnd	37	8	✔	✔	Tx/Fog	9BC0		B
PolygonOffsetFactor	37	9	✔	✔	Geometry	9BC8		B
PolygonOffsetBias	37	A	✔	✔	Geometry	9BD0		B
LineClipLengthThreshold	37	B	✔	✔	Geometry	9BD8		B
TriangleClipAreaThreshold	37	C	✔	✔	Cull	9BE0		B
RasterPosXIncrement	37	D	✔	✔		9BE8		B
RasterPosYIncrement	37	E	✔	✔		9BF0		B
UserClip0W	38	3	✔	✔	Geometry	9C00		B
UserClip3Z	38	E	✔	✔	Geometry	9C00		B
UserClip0X	38	0	✔	✔	Geometry	9C08		B
UserClip0Y	38	1	✔	✔	Geometry	9C10		B
UserClip3Y	38	D	✔	✔	Geometry	9C18		B
UserClip1W	38	7	✔	✔	Geometry	9C20		B
UserClip1X	38	4	✔	✔	Geometry	9C28		B
UserClip1Y	38	5	✔	✔	Geometry	9C30		B
UserClip0Z	38	2	✔	✔	Geometry	9C38		B
UserClip2W	38	B	✔	✔	Geometry	9C48		B
UserClip2X	38	8	✔	✔	Geometry	9C50		B

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
UserClip1Z	38	6	✔	✔	Geometry	9C58		B
UserClip2Z	38	A	✔	✔	Geometry	9C60		B
UserClip3W	38	F	✔	✔	Geometry	9C68		B
UserClip3X	38	C	✔	✔	Geometry	9C70		B
UserClip2Y	38	9	✔	✔	Geometry	9C78		B
UserClip4W	39	3	✔	✔	Geometry	9C88		B
UserClip4Y	39	1	✔	✔	Geometry	9C88		B
UserClip4X	39	0	✔	✔	Geometry	9C90		B
UserClip4Z	39	2	✔	✔	Geometry	9CA0		B
UserClip5W	39	7	✔	✔	Geometry	9CA8		B
UserClip5X	39	4	✔	✔	Geometry	9CB0		B
UserClip5Z	39	6	✔	✔	Geometry	9CB8		B
PackedSpecular	39	8		✔	Matrix	9CC0	int	B
UserClip5Y	39	5	✔	✔	Geometry	9CC0		B
PackedDiffuse	39	9		✔	Matrix	9CC8	int	B
PackedNormal	39	A		✔	Matrix	9CD0	int	B
PackedFaceNormal	39	B		✔	Matrix	9CD8	int	B
RasterPosXOffset	39	D	✔	✔		9CE8		B
RasterPosYOffset	39	E	✔	✔		9CF0		B
SceneAmbientColorRed	50	0	✔	✔	Material	A800		B
SceneAmbientColorGreen	50	1	✔	✔	Material	A808		B
SceneAmbientColorBlue	50	2	✔	✔	Material	A810		B
FrontAmbientColorRed	51	0	✔	✔	Material	A880		B
FrontAmbientColorGreen	51	1	✔	✔	Material	A888		B
FrontAmbientColorBlue	51	2	✔	✔	Material	A890		B
FrontDiffuseColorRed	51	3	✔	✔	Material	A898		B
FrontDiffuseColorGreen	51	4	✔	✔	Material	A8A0		B
FrontDiffuseColorBlue	51	5	✔	✔	Material	A8A8		B
FrontAlpha	51	6	✔	✔	Material	A8B0		B

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
FrontSpecularColorRed	51	7	✔	✔	Material	A8B8		B
FrontSpecularColorGreen	51	8	✔	✔	Material	A8C0		B
FrontSpecularColorBlue	51	9	✔	✔	Material	A8C8		B
FrontEmissiveColorRed	51	A	✔	✔	Material	A8D0		B
FrontEmissiveColorGreen	51	B	✔	✔	Material	A8D8		B
FrontEmissiveColorBlue	51	C	✔	✔	Material	A8E0		B
FrontSpecularExponent	51	D	✔	✔	Material	A8E8		B
FrontSpecularAlpha	51	E	✔	✔	Material	A8F0		B
BackAmbientColorRed	52	0	✔	✔	Material	A900		B
BackAmbientColorGreen	52	1	✔	✔	Material	A908		B
BackAmbientColorBlue	52	2	✔	✔	Material	A910		B
BackDiffuseColorRed	52	3	✔	✔	Material	A918		B
BackDiffuseColorGreen	52	4	✔	✔	Material	A920		B
BackDiffuseColorBlue	52	5	✔	✔	Material	A928		B
BackAlpha	52	6	✔	✔	Material	A930		B
BackSpecularColorRed	52	7	✔	✔	Material	A938		B
BackSpecularColorGreen	52	8	✔	✔	Material	A940		B
BackSpecularColorBlue	52	9	✔	✔	Material	A948		B
BackEmissiveColorRed	52	A	✔	✔	Material	A950		B
BackEmissiveColorGreen	52	B	✔	✔	Material	A958		B

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
BackEmissiveColorBlue	52	C	✔	✔	Material	A960		B
BackSpecularAlpha	52	E	✔	✔	Material	A970		B
DMAAddr	53	0	✕	✔	Command	A980	int	✕
DMACount	53	1	✕	✔	Command	A988	int	✕
Command Interrupt	53	2	✕	✔	Host In	A990	bitfield	✕
CommandInterrupt	53	2	✓	✓	Rectangle DMA	A990	bitfield	✕
DMARectangle Read	53	5	✕	✔	Rectangle DMA	A9A8	bitfield	✕
DMARectangleReadAddress	53	6	✔	✔	Rectangle DMA	A9B0	int	✕
DMARectangleReadLinePitch	53	7	✔	✔	Rectangle DMA	A9B8	int	✕
DMARectangleReadTarget	53	8	✔	✔	Rectangle DMA	A9C0	bitfield	✕
DMARectangleWrite	53	9	✕	✔	Rectangle DMA	A9C8	bitfield	✕
DMARectangleWriteAddress	53	A	✔	✔	Rectangle DMA	A9D0	int	✕
DMARectangleWriteLinePitch	53	B	✔	✔	Host In	A9D8	int	✕
DMAOutput Address	53	C	✕	✔	Rectangle DMA	A9E0	int	✕
DMAOutputCount	53	D	✕	✔	Rectangle DMA	A9E8	int	✕
DMAReadGLINTSource	53	E	✔	✔	Rectangle DMA	A9F0		✕
DMAContinue	53	F	✕	✔	Command	A9F8	int	✔
DMAFeedback	54	2	✕	✔	Rectangle DMA	AA10	int	✕
GeometryModeAnd	55	2	✕	✔	Geometry	AA90	bitfield	B
GeometryModeOr	55	3	✕	✔	Geometry	AA98	bitfield	B
MaterialModeOr	55	3	✕	✔	Material	AA98		B
NormaliseModeAnd	55	4	✕	✔	Normalisation	AAA0		B
NormaliseModeOr	55	5	✕	✔	Normalisation	AAA8		B
LightingModeAnd	55	6	✕	✔	Lighting	AAB0		B

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
LightingModeOr	55	7	✗	✔	Lighting	AAB8		B
ColorMaterialModeAnd	55	8	✗	✔	Material	AAC0		B
ColorMaterialModeOr	55	9	✗	✔	Material	AAC8		B
DeltaModeAnd	55	A	✗	✔	Delta	AAD0	bitfield	✗
DeltaModeOr	55	B	✗	✔	Delta	AAD8	bitfield	✗
PointModeAnd	55	C	✗	✔	Delta	AAE0		✗
PointModeOr	55	D	✗	✔	Delta	AAE8		✗
LineModeAnd	55	E	✗	✔	Delta	AAF0	bitfield	✗
LineModeOr	55	F	✗	✔	Delta	AAF8	bitfield	✗
TriangleModeAnd	56	0	✗	✔	Delta	AB00		✗
TriangleModeOr	56	1	✗	✔	Delta	AB08		✗
MaterialModeAnd	56	2	✗	✔	Material	AB10		B
PipeModeAnd	56	8	✗	✔	Pipe Manager	AB40		✗
PipeModeOr	56	9	✗	✔	Pipe Manager	AB48		✗
FogVertexModeAnd	56	A	✗	✔	Tx/Fog	AB50		B
FogVertexModeOr	56	B	✗	✔	Tx/Fog	AB58		B
TextureModeAnd	56	C	✗	✔	Delta	AB60		B
TextureModeOr	56	D	✗	✔	Delta	AB68		B
StripeFilterModeAnd	56	E	✗	✔	Cull	AB70		B
StripeFilterModeOr	56	F	✗	✔	Cull	AB78		B
WindowOr	57	1	✗	✔	Stencil	AB88	bitfield	✗
LBReadModeAnd	57	2	✗	✔	LB Read	AB90		✗
LBReadModeOr	57	3	✗	✔	LB Read	AB98		✗
RasterizerModeAnd	57	4	✗	✔	Rasterizer	ABA0	bitfield	✗
RasterizerModeOr	57	5	✗	✔	Rasterizer	ABA8	bitfield	✗
ScissorModeAnd	57	6	✗	✔	Scissor	ABB0	bitfield	✗
ScissorModeOr	57	7	✗	✔	Scissor	ABB8	bitfield	✗
LineStippleModeAnd	57	8	✗	✔	Stipple	ABC0	bitfield	✗
LineStippleModeOr	57	9	✗	✔	Stipple	ABC8	bitfield	✗
AreaStippleModeAnd	57	A	✗	✔	Stipple	ABD0	bitfield	✗

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
AreaStippleModeOr	57	B	×	✔	Stipple	ABD8	bitfield	×
ColorDDAModeAnd	57	C	×	✔	Color DDA	ABE0	bitfield	×
ColorDDAModeOr	57	D	×	✔	Color DDA	ABE8	bitfield	×
WindowAnd	57	D	×	✔	Stencil	ABE8	bitfield	×
AlphaTestModeAnd	57	E	×	✔	Alpha Blend & Alpha Test	ABF0	bitfield	×
AlphaTestModeOr	57	F	×	✔	Alpha Blend & Alpha Test	ABF8	bitfield	×
AntialiasModeAnd	58	0	×	✔	Alpha test	AC00	bitfield	×
AntialiasModeOr	58	1	×	✔	Alpha test	AC08	bitfield	×
FogModeAnd	58	2	×	✔	Fog	AC10	bitfield	×
FogModeOr	58	3	×	✔	Fog	AC18	bitfield	×
TextureCoordModeAnd	58	4	×	✔	Texture coord	AC20		×
TextureCoordModeOr	58	5	×	✔	Texture coord	AC28	bitfield	×
TextureReadMode0And	58	6	×	✔	Texture Read	AC30	bitfield	×
TextureReadMode0Or	58	7	×	✔	Texture Read	AC38	bitfield	×
TextureFormatAnd	58	8	5	4		AC40		×
TextureFormatOr	58	9	5	4		AC48		×
TextureApplicationModeAnd	58	A	×	✔	Texture Application	AC50	bitfield	×
TextureApplicationModeOr	58	B	×	✔	Texture Application	AC58	bitfield	×
StencilModeAnd	58	C	×	✔	Stencil	AC60	bitfield	×
StencilModeOr	58	D	×	✔	Stencil	AC68	bitfield	×
DepthModeAnd	58	E	×	✔	Depth	AC70	bitfield	×
DepthModeOr	58	F	×	✔	Depth	AC78	bitfield	×
LBWriteModeAnd	59	0	×	✔	LB Write	AC80	bitfield	×
LBWriteModeOr	59	1	×	✔	LB Write	AC88	bitfield	×
FBDestReadModeAnd	59	2	×	✔	FB Read	AC90	bitfield	×
FBDestReadModeOr	59	3	×	✔	FB Read	AC98	bitfield	×
FBSourceReadModeAnd	59	4	×	✔	FB Read	ACA0	bitfield	×

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
FBSourceReadModeOr	59	5	×	✔	FB Read	ACA8	bitfield	×
AlphaBlendColorModeAnd	59	6	×	✔	Alpha blend	ACB0	bitfield	×
AlphaBlendColorModeOr	59	7	×	✔	Alpha blend	ACB8	bitfield	×
ChromaTestModeAnd	59	8	×	✔	Color DDA & Alpha Blend	ACC0	bitfield	×
ChromaTestModeOr	59	9	×	✔	Color DDA & Alpha Blend	ACC8	bitfield	×
DitherModeAnd	59	A	×	✔	Dither	ACD0	bitfield	×
DitherModeOr	59	B	×	✔	Dither	ACD8	bitfield	×
LogicalOpModeAnd	59	C	×	✔	Logic Op	ACE0	bitfield	×
LogicalOpModeOr	59	D	×	✔	Logic Op	ACE8	bitfield	×
FBWriteModeAnd	59	E	×	✔	FB Write	ACF0	bitfield	×
FBWriteModeOr	59	F	×	✔	FB Write	ACF8	bitfield	×
FilterModeAnd	5A	0	×	✔	Host Out	AD00	bitfield	×
FilterModeOr	5A	1	×	✔	Host Out	AD08	bitfield	×
StatisticModeAnd	5A	2	×	✔	Host Out	AD10	bitfield	×
StatisticModeOr	5A	3	×	✔	Host Out	AD18	bitfield	×
FBDestReadEnablesAnd	5A	4	×	✔	FB Read	AD20	bitfield	×
FBDestReadEnablesOr	5A	5	×	✔	FB Read	AD28	bitfield	×
AlphaBlendAlphaModeAnd	5A	6	×	✔	Alpha blend	AD30	bitfield	×
AlphaBlendAlphaModeOr	5A	7	×	✔	Alpha blend	AD38	bitfield	×
TextureReadMode1And	5A	8	×	✔	Texture Read	AD40	bitfield	×
TextureReadMode1Or	5A	9	×	✔	Texture Read	AD48	bitfield	×
TextureFilterModeAnd	5A	A	×	✔	Texture	AD50	bitfield	×
TextureFilterModeOr	5A	B	×	✔	Texture	AD58	bitfield	×
LUTModeAnd	5A	E	×	✔	LUT	AD70	bitfield	×
LUTModeOr	5A	F	×	✔	LUT	AD78	bitfield	×

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
Zstart	5B	B	✔	✔	Fog	ADD8	int	×
dZdyDom	5B	C	✔	✔	Fog	ADE0		×
FBDestReadBufferAddr[0…3]	5D	0	✔	✔	FB Read	AE80	int	×
FBDestReadBufferOffset[0…3]	5D	4	✔	✔	FB Read	AEA0	int	×
FBDestReadBufferWidth[0…3]	5D	8	✔	✔	FB Read	AEC0	int	×
FBDestReadMode	5D	C	✔	✔	FB Read	AEE0	bitfield	×
FBDestReadEnables	5D	D	✔	✔	FB Read	AEE8	bitfield	×
FBSourceReadMode	5E	0	✔	✔	FB Read	AF00	bitfield	×
FBSourceReadBufferAddr	5E	1	✔	✔	FB Read	AF08	int	×
FBSourceReadBufferOffset	5E	2	✔	✔	FB Read	AF10	int	×
FBSourceReadBufferWidth	5E	3	✔	✔	FB Read	AF18	int	×
AlphaSourceColor	5F	0	✔	✔	Alpha blend	AF80	int	×
AlphaDestColor	5F	1	✔	✔	Alpha blend	AF88	bitfield	×
ChromaPassColor	5F	2	✔	✔	Color DDA & Alpha Blend	AF90	bitfield	×
ChromaFailColor	5F	3	✔	✔	Color DDA & Alpha Blend	AF98	bitfield	×
AlphaBlendColorMode	5F	4	✔	✔	Alpha blend	AFA0	bitfield	×
AlphaBlendAlphaMode	5F	5	✔	✔	Alpha blend	AFA8	bitfield	×
ConstantColorDDA	5F	6	×	✔	Color DDA	AFB0	bitfield	×
DeltaSwitchMode	5F	D	✔	✔	Delta Switch	AFE8	bitfield	×
DeltaSwitchModeAnd	5F	E	×	✔	Delta	AFF0	bitfield	×
DeltaSwitchModeOr	5F	F	×	✔	Delta	AFF8	bitfield	
FBWriteBufferAddr[0…3]	60	0	✔	✔	FB Write	B000	int	×
FBWriteBufferOffset[0…3]	60	4	✔	✔	FB Write	B020	int	×
FBWriteBufferWidth[0…3]	60	8	✔	✔	FB Write	B040	int	×

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
FBBlockColor[0…3]	60	C	✔	✔	FB Write	B060		×
FBBlockColorBack[0…3]	61	0	✔	✔	FB Write	B080		×
FBBlockColorBack	61	4	✔	✔	FB Write	B0A0		×
SizeOfFramebuffer	61	5	✔	✔	LB Read, FB Read, FB Write	B0A8	int	×
VTGAddress	61	6	✔	✔	FB Write	B0B0	int	✔
VTGData	61	7	×	✔	FB Write	B0B8	int	✔
ForegroundColor	61	8	✔	✔	Logic Ops	B0C0	int	×
BackgroundColor	61	9	✔	✔	Logic Ops	B0C8	int	×
DownloadAddress	61	A	✔	✔	FB Write	B0D0	int	×
DownloadData	61	B	×	✔	FB Write	B0D8		×
FogTable[0…15]	62	0	✔	✔	Fog	B100	bitfield	×
FogTable[16…31]	63	0	✔	✔	Fog	B180	bitfield	×
FogTable[32…47]	64	0	✔	✔	Fog	B200	bitfield	×
FogTable[48…63]	65	0	✔	✔	Fog	B280	bitfield	×
TextureCompositeMode	66	0	✔	✔	Texture Composite	B300	bitfield	×
TextureCompositeColor Mode0	66	1	✔	✔	Texture Composite	B308	bitfield	×
TextureCompositeAlpha Mode0	66	2	✔	✔	Texture Composite	B310	bitfield	×
TextureCompositeColorM ode1	66	3	✔	✔	Texture Composite	B318	bitfield	×
TextureCompositeAlpha Mode1	66	4	✔	✔	Texture Composite	B320		×
TextureCompositeFactor0	66	5	✔	✔	Texture Composite	B328	bitfield	
TextureCompositeFactor1	66	6	✔	✔	Texture Composite	B330	bitfield	×
TextureIndexMode0	66	7	✔	✔	Texture Index	B338	bitfield	×
TextureIndexMode1	66	8	✔	✔	Texture Index	B340	bitfield	×
LodRange0	66	9	✔	✔	Texture Index	B348	bitfield	×

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
LodRange1	66	A	✔	✔	Texture Index	B350	fixed	×
InvalidateCache	66	B	×	✔	Texture Read	B358	bitfield	✔
SetLogicalTexturePage	66	C	✔	✔	Texture Read	B360	bitfield	×
UpdateLogicalTextureInfo	66	D	×	✔	Texture Read	B368	tag	✔
TouchLogicalPage	66	E	×	✔	Texture Read	B370	bitfield	✔
LUTMode	66	F	✔	✔	LUT	B378	bitfield	×
TextureCompositeColor Mode0And	67	0	×	✔	Texture Composite	B380	bitfield	×
Mode0Or	67	1	×	✔	Texture Composite	B388	bitfield	×
TextureCompositeAlpha Mode0And	67	2	×	✔	Texture Composite	B390	bitfield	×
TextureCompositeAlpha Mode0Or	67	3	×	✔	Texture Composite	B398	bitfield	×
TextureCompositeColorM ode1And	67	4	×	✔	Texture Composite	B3A0	bitfield	×
TextureCompositeColorM ode1Or	67	5	×	✔	Texture Composite	B3A8	bitfield	×
TextureCompositeAlpha Mode1And	67	6	×	✔	Texture Composite	B3B0	bitfield	×
TextureCompositeAlpha Mode1Or	67	7	×	✔	Texture Composite	B3B8	bitfield	×
TextureIndexMode0And	67	8	×	✔	Texture Index	B3C0	bitfield	×
TextureIndexMode0Or	67	9	×	✔	Texture Index	B3C8	bitfield	×
TextureIndexMode1And	67	A	×	✔	Texture Index	B3D0	bitfield	×
TextureIndexMode1Or	67	B	×	✔	Texture Index	B3D8	bitfield	×
StencilDataAnd	67	C	×	✔	Stencil	B3E0	bitfield	×
StencilDataOr	67	D	×	✔	Stencil	B3E8	bitfield	×
TextureReadMode0	68	0	✔	✔	Texture Read	B400	bitfield	×
TextureReadMode1	68	1	✔	✔	Texture Read	B408	bitfield	×

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
TextureMapSize	68	5	✔	✔	Texture Read	B428	int	×
HeadPhysicalPageAllocation[0…3]	69	0	✔	✔	Texture Read	B480	int	×
TailPhysicalPageAllocation[0…3]	69	4	✔	✔	Texture Read	B4A0	int	×
PhysicalPageAllocationTableAddr	69	8	✔	✔	Texture Read	B4C0	int	×
BasePageOfWorkingSet	69	9	✔	✔	Texture Read	B4C8	int	×
LogicalTexturePageTableAddr	69	A	✔	✔	Texture Read	B4D0	int	×
LogicalTexturePageTableLength	69	B	✔	✔	Texture Read	B4D8	int	×
BasePageOfWorkingSetHost	69	C	✔	✔	Texture Read	B4E0	int	×
LBDestReadMode	6A	0	✔	✔	LB Read	B500	int	×
LBDestReadEnables	6A	1	✔	✔	LB Read	B508	bitfield	×
LBDestReadBufferAddr	6A	2	✔	✔	LB Read	B510	int	
LBDestReadBufferOffset	6A	3	✔	✔	LB Read	B518	int	
LBSourceReadMode	6A	4	✔	✔	LB Read	B520	int	×
LBSourceReadBufferAddr	6A	5	✔	✔	LB Read	B528	int	×
LBSourceReadBufferOffset	6A	6	✔	✔	LB Read	B530	bitfield	×
GIDMode	6A	7	✔	✔	LB Read	B538	bitfield	×
LBWriteBufferAddr	6A	8	✔	✔	LB Write	B540	int	×
LBWriteBufferOffset	6A	9	✔	✔	LB Write	B548	int	×
LBClearDataL	6A	A	✔	✔	LB Read	B550	int	×
LBClearDataU	6A	B	✔	✔	LB Read	B558	int	×
LBDestReadModeAnd	6B	0	×	✔	LB Read	B580	bitfield	×
LBDestReadModeOr	6B	1	×	✔	LB Read	B588	bitfield	×
LBDestReadEnablesAnd	6B	2	×	✔	LB Read	B590	bitfield	×
LBDestReadEnablesOr	6B	3	×	✔	LB Read	B598	bitfield	×

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
LBSourceReadModeAnd	6B	4	×	✔	LB Read	B5A0	bitfield	×
LBSourceReadModeOr	6B	5	×	✔	LB Read	B5A8	bitfield	×
GIDModeAnd	6B	6	×	✔	LB Read	B5B0	bitfield	×
GIDModeOr	6B	7	×	✔	LB Read	B5B8	bitfield	×
ReadMonitorModeAnd	6B	8	×	✔	ReadMonitor	B5C0	bitfield	
ReadMonitorModeOr	6B	9	×	✔	ReadMonitor	B5C8	bitfield	
RectanglePosition	6C	0	✔	✔	2D Set Up	B600	int	×
GlyphPosition	6C	1	✔	✔	2D Set Up	B608	int	×
RenderPatchOffset	6C	2	✔	✔	2D Set Up	B610	bitfield	×
Config2D	6C	3	×	✔	Global	B618	bitfield	×
Packed8Pixels	6C	6	×	✔	2D Set Up	B630	int	✔
Packed16Pixels	6C	7	×	✔	2D Set Up	B638	int	✔
Render2D	6C	8	×	✔	2D Set Up	B640	bitfield	×
Render2DGlyph	6C	9	×	✔	2D Set Up	B648	bitfield	×
DownloadTarget	6C	A	✔	✔	2D Set Up	B650		✔
DownloadGlyphWidth	6C	B	✔	✔	2D Set Up	B658	int	×
GlyphData	6C	C	×	✔	2D Set Up	B660	int	×
Packed4Pixels	6C	D	×	✔	2D Set Up	B668	int	✔
RLData	6C	E	✔	✔	2D Set Up	B670	int	×
RLCount	6C	F	×	✔	2D Set Up	B678	int	×
IndexBaseAddress	6E	0	✔	✔	Host In	B700	int	×
VertexBaseAddress	6E	1	✔	✔	Host In	B708	int	×
IndexedTriangleList	6E	2	×	✔	Host In	B710	int	×
IndexedTriangleFan	6E	3	×	✔	Host In	B718	int	×
IndexedTriangleStrip	6E	4	×	✔	Host In	B720	int	×
IndexedLineList	6E	5	×	✔	Host In	B728	int	×
IndexedLineStrip	6E	6	×	✔	Host In	B730	int	×
IndexedPointList	6E	7	×	✔	Host In	B738	int	×
IndexedPolygon	6E	8	×	✔	Host In	B740	int	×
VertexTriangleList	6E	9	×	✔	Host In	B748	int	×
VertexTriangleFan	6E	A	×	✔	Host In	B750	int	×

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
VertexTriangleStrip	6E	B	✗	✔	Host In	B758	int	✗
VertexLineList	6E	C	✗	✔	Host In	B760	int	✗
VertexLineStrip	6E	D	✗	✔	Host In	B768	int	✗
VertexPointList	6E	E	✗	✔	Host In	B770	int	✗
VertexPolygon	6E	F	✗	✔	Host In	B778	int	✗
VertexValid	6F	1	✔	✔	Host In	B788	int	✗
VertexFormat	6F	2	✔	✔	Host In	B790	int	✗
VertexControl	6F	3	✔	✔	Host In	B798	bitfield	✗
RetainedRender	6F	4	✔	✔	Host In	B7A0	bitfield	✔
IndexedVertex	6F	5	✗	✔	Host In	B7A8	int	✗
IndexedDoubleVertex	6F	6	✗	✔	Host In	B7B0	int	✗
Vertex0	6F	7	✗	✔	Host In	B7B8	int	✗
Vertex1	6F	8	✗	✔	Host In	B7C0	int	✗
Vertex2	6F	9	✗	✔	Host In	B7C8	int	✗
VertexData0	6F	A	✗	✔	Host In	B7D0	int	✗
VertexData1	6F	B	✗	✔	Host In	B7D8	int	✗
VertexData2	6F	C	✗	✔	Host In	B7E0	int	✗
VertexData	6F	D	✗	✔	Host In	B7E8	int	✗
VertexTagList[0…15]	70	0	✔	✔	Host In	B800	bitfield	✗
VertexTagList[16…31]	71	0	✔	✔	Host In	B880	bitfield	✗
SetPipe	81	0	✔	✔	Pipe Manager	C080	bitfield	✗
RestartPipe	81	1	✗	✔	Pipe Manager	C088	bitfield	✗
TAq	87	0	✗	✔	Matrix	C380		✗
TAr	87	1	✗	✔	Matrix	C388		✗
TAt	87	2	✗	✔	Matrix	C390		✗
TAs1	87	3	✗	✔	Matrix	C398		✗
TAs2	87	4	✗	✔	Matrix	C3A0		✗
TAs3	87	5	✗	✔	Matrix	C3A8		✗
TAs4	87	6	✗	✔	Matrix	C3B0		✗
TAs3q	87	7	✗	✔	Matrix	C3B8		✗
TBq	87	8	✗	✔	Matrix	C3C0		✗
TBr	87	9	✗	✔	Matrix	C3C8		✗

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
TBt	87	A	×	✔	Matrix	C3D0		×
TBs1	87	B	×	✔	Matrix	C3D8		×
TBs2	87	C	×	✔	Matrix	C3E0		×
TBs3	87	D	×	✔	Matrix	C3E8		×
TBs4	87	E	×	✔	Matrix	C3F0		×
TBs3q	87	F	×	✔	Matrix	C3F8		×
TCq	88	0	×	✔	Matrix	C400		×
TEq	88	0	×	✔	Matrix	C400		×
TCr	88	1	×	✔	Matrix	C408		×
TCt	88	2	×	✔	Matrix	C410		×
TCs1	88	3	×	✔	Matrix	C418		×
TCs2	88	4	×	✔	Matrix	C420		×
TCs3	88	5	×	✔	Matrix	C428		×
TCs4	88	6	×	✔	Matrix	C430		×
TCs3q	88	7	×	✔	Matrix	C438		×
TDq	88	8	×	✔	Matrix	C440		×
TDr	88	9	×	✔	Matrix	C448		×
TDt	88	A	×	✔	Matrix	C450		×
TDs1	88	B	×	✔	Matrix	C458		×
TDs2	88	C	×	✔	Matrix	C460		×
TDs3	88	D	×	✔	Matrix	C468		×
TDs4	88	E	×	✔	Matrix	C470		×
TDs3q	88	F	×	✔	Matrix	C478		×
TEr	89	1	×	✔	Matrix	C488		×
TEt	89	2	×	✔	Matrix	C490		×
TEs1	89	3	×	✔	Matrix	C498		×
TEs2	89	4	×	✔	Matrix	C4A0		×
TEs3	89	5	×	✔	Matrix	C4A8		×
TEs4	89	6	×	✔	Matrix	C4B0		×
TEs3q	89	7	×	✔	Matrix	C4B8		×
TFq	89	8	×	✔	Texture Fog	C4C0		×
TFr	89	9	×	✔	Texture Fog	C4C8		×
TFt	89	A	×	✔	Texture Fog	C4D0		×
TFs1	89	B	×	✔	Texture Fog	C4D8		×
TFs2	89	C	×	✔	Texture Fog	C4E0		×
TFs3	89	D	×	✔	Texture Fog	C4E8		×
TFs4	89	E	×	✔	Texture Fog	C4F0		×

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
TFs3q	89	F	✗	✔	Texture Fog	C4F8		✗
TGq	8A	0	✗	✔	Texture Fog	C500		✗
TGr	8A	1	✗	✔	Texture Fog	C508		✗
TGt	8A	2	✗	✔	Texture Fog	C510		✗
TGs1	8A	3	✗	✔	Texture Fog	C518		✗
TGs2	8A	4	✗	✔	Texture Fog	C520		✗
TGs3	8A	5	✗	✔	Texture Fog	C528		✗
TGs4	8A	6	✗	✔	Texture Fog	C530		✗
TGs3q	8A	7	✗	✔	Texture Fog	C538		✗
THq	8A	8	✗	✔	Texture Fog	C540		✗
THr	8A	9	✗	✔	Texture Fog	C548		✗
THt	8A	A	✗	✔	Texture Fog	C550		✗
THs1	8A	B	✗	✔	Texture Fog	C558		✗
THs2	8A	C	✗	✔	Texture Fog	C560		✗
THs3	8A	D	✗	✔	Texture Fog	C568		✗
THs4	8A	E	✗	✔	Texture Fog	C570		✗
THs3q	8A	F	✗	✔	Texture Fog	C578		✗
StripeOwnership	8B	0	✔	✔	Cull	C580		✗B
StripeFilterOffset	8B	1	✔	✔	Cull	C588		✗B
PointExtend	8B	2	✔	✔		C590		✗B
AAPointExtend	8B	3	✔	✔	Cull	C598		✗B
LineExtend	8B	4	✔	✔	Cull	C5A0		✗B
AALineExtend	8B	5	✔	✔	Cull	C5A8		✗B
TriangleExtend	8B	6	✔	✔	Cull	C5B0		✗B
AATriangleExtend	8B	7	✔	✔	Cull	C5B8		✗B
StripeYBias	8B	8	✔	✔	Cull	C5C0		✗B
TBFactor0	8B	A	✔	✔	Matrix	C5D0		✗B
TBFactor1	8B	B	✔	✔	Matrix	C5D8		✗B
NBFactor0	8B	E	✔	✔	Normalisation	C5F0		✗B
NBFactor1	8B	F	✔	✔	Normalisation	C5F8		✗
MatrixModeAnd	8C	0	✗	✔	Matrix	C600		✗
MatrixModeOr	8C	1	✗	✔	Matrix	C608		✗
PipeLoadAnd	8C	2	✗	✔	Pipe Manager	C610		✗
PipeLoadOr	8C	3	✗	✔	Pipe Manager	C618		✗
MatrixSelect	8D	0	✔	✔	Matrix	C680		✗

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
MatrixStatus	8D	1	✔	✔	Matrix	C688		×
SetUnitMatrix	8D	2	×	✔		C690		×
InvertCurrentMatrix	8D	9	×	✔		C6C8		✔
PipeLoad	8D	C	×	✔	Pipe Manager	C6E0		×
BoundingBoxMinX	8F	A	✔	✔	Matrix	C7D0		×
BoundingBoxMinY	8F	B	✔	✔	Matrix	C7D8		×
BoundingBoxMinZ	8F	C	✔	✔	Matrix	C7E0		×
BoundingBoxMaxX	8F	D	✔	✔	Matrix	C7E8		×
BoundingBoxMaxY	8F	E	✔	✔	Matrix	C7F0		×
BoundingBoxMaxZ	8F	F	✔	✔	Matrix	C7F8		×
StartBoundingVolume	91	0	✓	✓		C880		×
EndBoundingVolume	91	1	×	✓	Matrix	C888		×
BoundingVertexX	91	3	✓	✓	Matrix	C898		×
BoundingVertexY	91	4	✓	✓	Matrix	C8A0		×
BoundingVertexZ	91	5	✓	✓	Matrix	C8A8		×
BNz	92	0	✓	✓	Matrix	C900		×
BNy	92	1	✓	✓	Matrix	C908		×
BNx	92	2	✓	✓	Matrix	C910		×
BVw	92	3	✓	✓	Matrix	C918		×
BVz	92	4	✓	✓		C920		×
BVy	92	5	✓	✓		C928		×
BVx2	92	6	✓	✓	Matrix	C930		×
BVx3	92	7	✓	✓	Matrix	C938		×
BVx4	92	8	✓	✓	Matrix	C940		×
CommandDMAControl	94	0	✓	✓		CA00		×
CommandDMAControlAnd	94	1	×	✓		CA08		×
CommandDMAControlOr	94	2	×	✓		CA10		×
DMAFailAddr	94	3	✓	✓	Command	CA18		×
DMAFailCount	94	4	✓	✓	Command	CA20		×
BoundingBoxTest	94	5	×	✓	Matrix	CA28		✔

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
BoundingVolumeTest	94	6	×	✓	Matrix	CA30		×
DMABoundingBox Jump	94	7	×	✓	Command	CA38		×
Scratch	94	8	✓	✓		CA40		×
DMARectangleRead Control	94	9	✓	✓	Rectangle DMA	CA48		×
DMARectangleRead ControlAnd	94	A	×	✓	Rectangle DMA	CA50		×
DMARectangleRead ControlOr	94	B	×	✓	Rectangle DMA	CA58		×
DMARectangleReadID	94	C	✓	✓	Rectangle DMA	CA60		×
DMARectangleWriteContrl	94	D	✓	✓	Rectangle DMA	CA68		×
DMARectangleWriteControlAnd	94	E	×	✓	Rectangle DMA	CA70		×
DMARectangleWriteControlOr	94	F	×	✓	Rectangle DMA	CA78		×
CommandID[0..15]	95	0	✓	✓		CA80		×
ArrayFormatLoad[0,1]	97	0	✓	✓	Vertex Array	CB80		×
ArrayFormatValid[0,1]	97	2	✓	✓	Vertex Array	CB90		×
ArrayFormatRef[0..7]	97	4	✓	✓	Vertex Array	CBA0		×
ArrayFormatField[0..7]	97	C	✓	✓	Vertex Array	CBE0		×
ArrayFormatLoadAnd[0,1]	98	0	×	✓	Vertex Array	CC00		×
ArrayFormatValidAnd[0,1]	98	2	×	✓	Vertex Array	CC10		×
ArrayFormatRefAnd[0..7]	98	4	×	✓	Vertex Array	CC20		×
ArrayFormatFieldAnd[0..7]	98	C	×	✓	Vertex Array	CC60		×
ArrayFormatLoadOr[0,1]	99	0	×	✓	Vertex Array	CC80		×

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
ArrayFormatValidOR[0,1]	99	2	×	✓	Vertex Array	CC90		×
ArrayFormatRefOr[0..7]	99	4	×	✓	Vertex Array	CCA0		×
ArrayFormatFieldOr[0..7]	99	C	×	✓	Vertex Array	CCE0		×
ArrayTagTable[0..15]	9A	0	✓	✓	Vertex Array	CD00		×
ArrayControl	9B	0	✓	✓	Vertex Array	CD80		×
ArrayEnable	9B	1	✓	✓	Vertex Array	CD88		×
ArrayControlAnd	9B	2	×	✓	Vertex Array	CD90		×
ArrayEnableAnd	9B	3	×	✓	Vertex Array	CD98		×
ArrayControlOr	9B	4	×	✓	Vertex Array	CDA0		×
ArrayEnableOr	9B	5	×	✓	Vertex Array	CDA8		×
ArrayMode	9B	6	✓	✓	Vertex Array	CDB0		×
ArrayModeAnd	9B	7	×	✓	Vertex Array	CDB8		×
ArrayModeOr	9B	8	×	✓	Vertex Array	CDC0		×
ArrayVertexLoadMask0	9B	9	✓	✓	Vertex Array	CDC8		×
ArrayVertexLoadMask1	9B	A	✓	✓	Vertex Array	CDD0		×
ArrayDataOffset	9C	1	✓	✓	Vertex Array	CE08		×
ArrayDataSingle	9C	2	×	✓	Vertex Array	CE10		×
ArrayDataMultiple	9C	3	×	✓	Vertex Array	CE18		×
ArrayIndexSingle	9C	4	×	✓	Vertex Array	CE20		×
ArrayDataLoad	9C	7	✓	✓	Vertex Array	CE38		×
CommandID	9C	E	✓	✓	Vertex Array	CE70		×
CommandFilter	9C	F	✓	✓		CE78		×
ArrayVertexStoreValid	9D	1	×	×	Vertex Array	CE88	bitfiled	×
DataSync	9D	8	✓	✓		CEC0		×
DataSyncTarget	9D	9	✓	✓		CEC8		×
TMask	9D	A	✓	✓	Texture Fog	CED0		×
DeltaModeOverride0	9D	B	✓	✓		CED8		×
DeltaModeOverride1	9D	C	✓	✓		CEE0		×

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
ArrayVertexStore[0-63]	9E	0	✔	✔	Vertex Array	CF00	bitfiled	×
ArrayIndexAddress[0..15]	A2	0	✓	✓	Vertex Array	D100		×
ArrayIndexSize	A3	0	✓	✓	Vertex Array	D180		×
ArrayDataAddress[0..15]	A4	0	✓	✓	Vertex Array	D200		×
ArrayDataSize[0..15]	A5	0	✓	✓	Vertex Array	D280		×
ArrayIndex[0..15]	A6	0	✓	✓	Vertex Array	D300		×
ArrayDataIndex[0..15]	A7	0	✓	✓	Vertex Array	D380		×
ArrayDataID[0..15]	A9	0	✓	✓	Vertex Array	D480		×
DMACurrent Address[0..15]	AA	0	✓	✓	Command	D500		×
LightMode	AB	0	✓	✓	Lighting	D580		×B
LightAmbientIntensityRed	AB	1	✓	✓	Lighting	D588		×B
LightAmbientIntensityGreen	AB	2	✓	✓	Lighting	D590		×B
LightAmbientIntensityBlue	AB	3	✓	✓	Lighting	D598		×B
LightDiffuseIntensityRed	AB	4	✓	✓	Lighting	D5A0		×B
LightDiffuseIntensityGreen	AB	5	✓	✓	Lighting	D5A8		×B
LightDiffuseIntensityBlue	AB	6	✓	✓	Lighting	D5B0		×B
LightSpecularIntensityRed	AB	7	✓	✓	Lighting	D5B8		×B
LightSpecularIntensityGreen	AB	8	✓	✓	Lighting	D5C0		×B
LightSpecularIntensityBlue	AB	9	✓	✓	Lighting	D5C8		×B
LightPositionX	AB	A	✓	✓	Lighting	D5D0		×B
LightPositionY	AB	B	✓	✓	Lighting	D5D8		×B
LightPositionZ	AB	C	✓	✓	Lighting	D5E0		×B
LightPositionW	AB	D	✓	✓	Lighting	D5E8		×B
LightSpotlightDirectionX	AB	E	✓	✓	Lighting	D5F0		×B

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
LightSpotlightDirectionY	AB	F	✓	✓	Lighting	D5F8		✕B
LightSpotlightDirectionZ	AC	0	✓	✓	Lighting	D600		✕B
LightSpotlightExponent	AC	1	✓	✓	Lighting	D608		✕B
LightCosSpotlightCutoffAngleOuter	AC	2	✓	✓	Lighting	D610		✕B
LightConstantAttenuation	AC	3	✓	✓	Lighting	D618		✕B
LightLinearAttenuation	AC	4	✓	✓	Lighting	D620		✕B
LightQuadraticAttenuation	AC	5	✓	✓	Lighting	D628		✕B
LightCosSpotlightCutoffAngleInner	AC	6	✓	✓	Lighting	D630		✕B
LightCosSpotlightInvSlope	AC	7	✓	✓	Lighting	D638		✕B
LightRange	AC	8	✓	✓	Lighting	D640		✕B
TextureFormatControl	AD	0	✓	✓	Texture	D680	bitfield	✕B
TextureFormatControlAnd	AD	1	✕	✓	Texture	D688		✕B
TextureFormatControlOr	AD	2	✕	✓	Texture	D690		✕B
TextureLODScaleA	AD	3	✓	✓		D698		✕B
TextureLODScaleB	AD	4	✓	✓		D6A0		✕B
PackedBV	B4	0	✕	✓	Matrix	DA00	int	✕B
PackedN	B4	1	✕	✓	Matrix	DA08	int	✕B
PackedFN	B4	2	✕	✓	Matrix	DA10	int	✕B
PackedTA	B4	3	✕	✓	Matrix	DA18	int	✕B
PackedTB	B4	4	✕	✓	Matrix	DA20	int	✕B
PackedTC	B4	5	✕	✓	Matrix	DA28	int	✕B
PackedTD	B4	6	✕	✓	Matrix	DA30	int	✕B
PackedTE	B4	7	✕	✓	Matrix	DA38	int	✕B
PackedTF	B4	8	✕	✓	Matrix	DA40	int	✕B
PackedTG	B4	9	✕	✓	Matrix	DA48	int	✕B
PackedTH	B4	A	✕	✓	Matrix	DA50	int	✕B

Name	Group	Local Offset	Read	Write	Unit	Global Offset	Format	Command/ control/ broadcast
PackedV	B4	F	×	✓	Matrix	DA78	int	×B

INDEX