



***Matrox Graphics Inc.***

***Matrox  
MGA-2064W Specification***

---

***Special Internet Edition***

***Document Number 10470-MS-0300***

***Feb. 29, 1996***



## Trademark Acknowledgements

*MGA,<sup>TM</sup> MGA-2064W,<sup>TM</sup> MGA-1164SG,<sup>TM</sup> MGA-2064W,<sup>TM</sup> MGA-2164W,<sup>TM</sup> MGA-VC064SFB,<sup>TM</sup> MGA-VC164SFB,<sup>TM</sup> MGA Marvel,<sup>TM</sup> MGA Millennium,<sup>TM</sup> MGA Mystique,<sup>TM</sup> MGA Rainbow Runner,<sup>TM</sup> MGA DynaView,<sup>TM</sup> PixelTOUCH,<sup>TM</sup> MGA Control Panel,<sup>TM</sup> and Instant ModeSWITCH,<sup>TM</sup> are trademarks of Matrox Graphics Inc.*

*Matrox<sup>®</sup> is a registered trademark of Matrox Electronic Systems Ltd.*

*VGA,<sup>®</sup> is a registered trademark of International Business Machines Corporation; Micro Channel<sup>TM</sup> is a trademark of International Business Machines Corporation.*

*Intel<sup>®</sup> is a registered trademark, and 386,<sup>TM</sup> 486,<sup>TM</sup> Pentium,<sup>TM</sup> and 80387<sup>TM</sup> are trademarks of Intel Corporation.*

*Windows<sup>TM</sup> is a trademark of Microsoft Corporation; Microsoft,<sup>®</sup> and MS-DOS<sup>®</sup> are registered trademarks of Microsoft Corporation.*

*AutoCAD<sup>®</sup> is a registered trademark of Autodesk Inc.*

*RAMDAC<sup>TM</sup> is a trademark of Brooktree.*

*Unix<sup>TM</sup> is a trademark of AT&T Bell Laboratories.*

*X-Windows<sup>TM</sup> is a trademark of the Massachusetts Institute of Technology.*

*AMD<sup>TM</sup> is a trademark of Advanced Micro Devices. Atmel<sup>®</sup> is a registered trademark of Atmel Corporation. Catalyst<sup>TM</sup> is a trademark of Catalyst Semiconductor Inc. SGS<sup>TM</sup> is a trademark of SGS-Thompson. Toshiba<sup>TM</sup> is a trademark of Toshiba Corporation. Texas Instruments<sup>TM</sup> is a trademark of Texas Instruments. National<sup>TM</sup> is a trademark of National Semiconductor Corporation. Microchip<sup>TM</sup> is a trademark of Microchip Technology Inc.*

*All other nationally and internationally recognized trademarks and tradenames are hereby acknowledged.*

***This document contains confidential proprietary information that may not be disclosed without written permission from Matrox Graphics Inc.***

*© Copyright Matrox Graphics Inc., 1994, 1995, 1996. All rights reserved.*

*Disclaimer: Matrox Graphics Inc. reserves the right to make changes to specifications at any time and without notice. The information provided by this document is believed to be accurate and reliable. However, no responsibility is assumed by Matrox Graphics Inc. for its use; nor for any infringements of patents or other rights of third parties resulting from its use. No license is granted under any patents or patent rights of Matrox Graphics Inc. or Matrox Electronic Systems Ltd.*

# Contents

---

---

## Chapter 1: MGA Overview

1.1 Introduction . . . . .	1-2
1.2 MGA-2064W Typical Implementation . . . . .	1-2
1.3 Typographical Conventions . . . . .	1-3

## Chapter 2: MGA-2064W Overview

2.1 Introduction . . . . .	2-2
2.2 PCI Bus interface . . . . .	2-2
2.3 VGA Graphics Controller . . . . .	2-2
2.4 VGA Attributes Controller . . . . .	2-2
2.5 CRTC . . . . .	2-2
2.6 Address Processing Unit (APU) . . . . .	2-4
2.7 Data Processing Unit (DPU) . . . . .	2-4
2.8 Frame Buffer . . . . .	2-4

## Chapter 3: Resource Mapping

3.1 Memory Mapping . . . . .	3-2
3.1.1 Configuration Space Mapping . . . . .	3-2
3.2 Memory Space Mapping . . . . .	3-3
3.2.1 MGA General Map . . . . .	3-3
3.2.2 MGA Control Aperture Detailed Map . . . . .	3-3
3.2.3 Register Map . . . . .	3-4
3.2.4 I/O Space Mapping . . . . .	3-8
3.3 VGA Register Index Summary . . . . .	3-10

## Chapter 4: Register Descriptions

4.1 Power Graphic Mode Register Descriptions . . . . .	4-2
4.1.1 Power Graphic Mode Configuration Space Registers . . . . .	4-2
4.1.2 Power Graphic Mode Memory Space Registers . . . . .	4-16
4.2 VGA Mode Registers . . . . .	4-77

## Chapter 5: Programmer's Specification

5.1 PCI Interface . . . . .	5-2
5.1.1 Direct Access Read Cache . . . . .	5-2
5.1.2 Big Endian Support . . . . .	5-2
5.1.3 Host Pixel Format . . . . .	5-6
5.2 WRAM Interface . . . . .	5-10
5.2.1 Frame Buffer Organization . . . . .	5-10
5.2.1.1 VGA mode . . . . .	5-10
5.2.1.2 Power Graphic Mode (non-interleave) . . . . .	5-11
5.2.1.3 Power Graphic Mode (interleave) . . . . .	5-11

5.2.2	Pixel Format	5-11
5.3	Chip Configuration and Initialization	5-13
5.3.1	Reset	5-13
5.3.2	Power Up Sequence	5-14
5.3.3	Operation Mode Selection	5-15
5.4	Direct Frame Buffer Access	5-15
5.5	Drawing in Power Graphic Mode	5-16
5.5.1	Overview	5-16
5.5.2	Global Initialization (all operations)	5-16
5.5.3	Line Programming	5-17
5.5.3.1	Slope Initialization	5-17
5.5.3.2	Solid Lines	5-18
5.5.3.3	Lines That Use a Linestyle	5-19
5.5.3.4	Lines with Depth	5-20
5.5.4	Trapezoid / Rectangle Fill Programming	5-21
5.5.4.1	Slope Initialization	5-21
5.5.4.2	Constant Shaded Trapezoids / Rectangle Fills	5-23
5.5.4.3	Patterned Trapezoids / Rectangle Fills	5-24
5.5.4.4	Gouraud Shaded Trapezoids / Rectangle Fills	5-26
5.5.4.5	Textured Trapezoids / Rectangle Fills	5-27
5.5.5	Bitblt Programming	5-28
5.5.5.1	Address Initialization	5-28
5.5.6	Two-operand Bitblts	5-29
5.5.6.1	Two-operand Fast Bitblts	5-30
5.5.6.2	Color Patterning 8 x 8	5-31
5.5.6.3	BitBlts With Expansion (Character Drawing) 1 bpp	5-33
5.5.6.4	BitBlts With Expansion (Character Drawing) 1 bpp Planar	5-34
5.5.7	ILOAD Programming	5-35
5.5.7.1	Address Initialization	5-36
5.5.7.2	ILOAD of Two-operand Bitblt	5-37
5.5.7.3	ILOAD with Expansion (Character Drawing)	5-39
5.5.8	Scaling	5-40
5.5.8.1	Scaling Initialization	5-42
5.5.9	IDUMP Programming	5-43
5.6	CRTC Programming	5-45
5.6.1	Horizontal Timing	5-45
5.6.2	Vertical Timing	5-46
5.6.3	Memory Address Counter	5-47
5.6.4	Programming in VGA Mode	5-48
5.6.5	Programming in Power Graphic Mode	5-49
5.7	Interrupt Programming	5-53

## Alphabetical List of Register Fields / Index

# *List of Figures*

---

---

## **Chapter 1: MGA Overview**

Figure 1-1: Typical Implementation Block Diagram ..... 1-2

## **Chapter 2: MGA-2064W Overview**

Figure 2-1: MGA-2064W Block Diagram ..... 2-3

## **Chapter 5: Programmer's Specification**

Figure 5-1: Drawing Multiple Primitives ..... 5-21

Figure 5-2: CRTIC Horizontal Timing ..... 5-45

Figure 5-3: CRTIC Vertical Timing ..... 5-46

Figure 5-4: Video Timing in Interlace Mode ..... 5-52



# ***List of Tables***

---

---

## **Chapter 3: Resource Mapping**

Table 3-1: MGA-2064W Configuration Space Mapping . . . . .	3-2
Table 3-2: MGA General Map . . . . .	3-3
Table 3-3: MGA Control Aperture Detailed Map . . . . .	3-3
Table 3-4: Register Mapping . . . . .	3-4
Table 3-5: I/O Space Mapping. . . . .	3-8
Table 3-6: VGA Indices . . . . .	3-10

## **Chapter 5: Programmer's Specification**

Table 5-1: Fast Bitblt Alignment Constraints . . . . .	5-30
Table 5-2: ILOAD Source Size. . . . .	5-36
Table 5-3: ILOAD Supported Formats . . . . .	5-38
Table 5-4: Bitblt with Expansion Supported Formats. . . . .	5-39
Table 5-5: Scaling Supported Formats . . . . .	5-40
Table 5-6: Scaling Source Size . . . . .	5-41
Table 5-7: IDUMP Source Size . . . . .	5-44
Table 5-8: IDUMP Supported Formats . . . . .	5-44







## ***Chapter 1: MGA Overview***

*This chapter includes:*

Introduction .....	1-2
MGA-2064W Typical Implementation .....	1-2
Typographical Conventions .....	1-3

## 1.1 Introduction

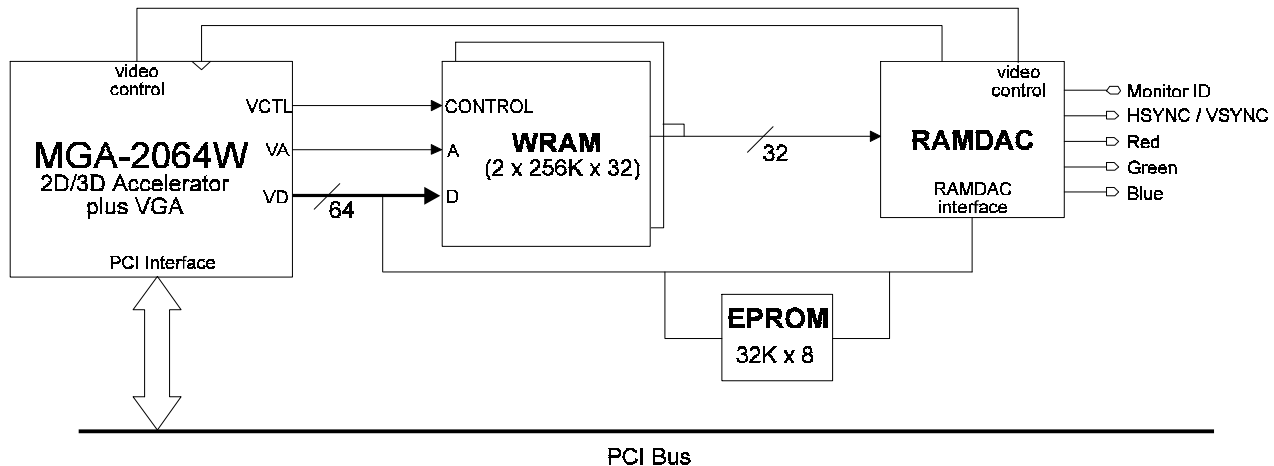
Matrox MGA is a high-speed, high-resolution graphics accelerator series of products designed for the power user. MGA is very suitable for GUI environments such as Microsoft Windows 3.1, Chicago, Windows NT, IBM OS/2 PM, Unix X-Windows, and AutoCAD. It offers ultra high resolution displays with true color, real-time 3D, Video for Windows acceleration, and many other innovative hardware and software enhancements.

MGA's 64-bit graphics power, in combination with a 486, PowerPC, or Pentium-class PC is in our opinion the best graphics solution if you require true workstation-level performance at a reasonable price.

## 1.2 MGA-2064W Typical Implementation

The MGA-2064W chip lies at the heart of MGA's powerful graphics subsystem. Several possible memory configurations permit design of 8,16, 24, and 32 bits/pixel displays at resolutions up to 1600 x 1200 pixels. [Figure 1-1](#) shows a block diagram of a typical graphics display adapter which uses the MGA-2064W chip.

*Figure 1-1: Typical Implementation Block Diagram*



A small number of devices is required to implement the graphics subsystem:

- MGA-2064W chip
- 2 WRAMs
- RAMDAC
- EPROM (this device is optional)

The possibility exists for other implementations that use more memory and a different RAMDAC to create combinations of higher resolutions and greater pixel depths. With the addition of glue logic, it is also possible to interface to system buses other than PCI (such as ISA, VESA VL, EISA, Micro Channel, and so on).

## 1.3 Typographical Conventions

<i>Description</i>	<i>Example</i>				
Active low signals are indicated by a trailing forward slash. Signal names appear in upper-case characters.	VHSYNC/				
Numbered signals appear within angle brackets, separated by a colon.	MA<8:0>				
Register names are indicated by upper-case bold sans-serif letters.	<b>DEVID</b>				
Fields within registers are indicated by lower-case bold sans-serif letters.	<b>vendor</b>				
Bits within a field appear within angle brackets, separated by a colon.	<b>vendor</b> <15:0>				
Hexadecimal values are indicated by a trailing letter 'h'.	CFFFh				
Binary values are indicated by a trailing letter 'b'.	0000 0010b				
Register description reset values that are underlined are reset on soft reset.	00 <u>00</u> 0101				
An 'X' in a register value indicates that that bit is undefined.	0000 01XX				
Emphasized text and table column titles are set in bold italics.	This bit <i><b>must be set.</b></i>				
In the register description pages in Chapter 4, when a description is continued from another page, the page heading contains ellipses (...) to indicate this.	<b>... OPMODE</b>				
In <a href="#">Chapter 5</a> 's <b>DWGCTL</b> illustrations, the '+' and '#' symbols have a special meaning. This is explained in ' <a href="#">Overview</a> ' on page 5-16.	<p><b>trans</b></p> <table border="1"> <tr> <td>#</td> <td>#</td> <td>#</td> <td>#</td> </tr> </table>	#	#	#	#
#	#	#	#		





## ***Chapter 2: MGA-2064W Overview***

*This chapter includes:*

Introduction .....	2-2
PCI Bus interface .....	2-2
VGA Graphics Controller .....	2-2
VGA Attributes Controller .....	2-2
CRTC .....	2-2
Address Processing Unit (APU) .....	2-4
Data Processing Unit (DPU) .....	2-4
Frame Buffer .....	2-4

## 2.1 Introduction

The MGA-2064W chip is a stand-alone graphics controller which is composed of several sections that work together to accomplish the tasks that are required of them. The individual sections of the MGA-2064W chip are listed below and described in detail in the remainder of this chapter.

- PCI bus interface
- VGA graphics controller
- VGA attributes controller
- CRTC
- Address Processing Unit (APU)
- Data Processing Unit (DPU)
- Frame buffer

## 2.2 PCI Bus interface

This section of the MGA-2064W chip implements the interface with the host processor. It includes:

- All of the decoding circuitry for the PCI interface
- Decoding of all resources
- Configuration registers

## 2.3 VGA Graphics Controller

This section of the MGA-2064W implements the VGA-compatible access to the frame buffer. This section includes:

- Graphics controller registers
- Data path between the host and the frame buffer

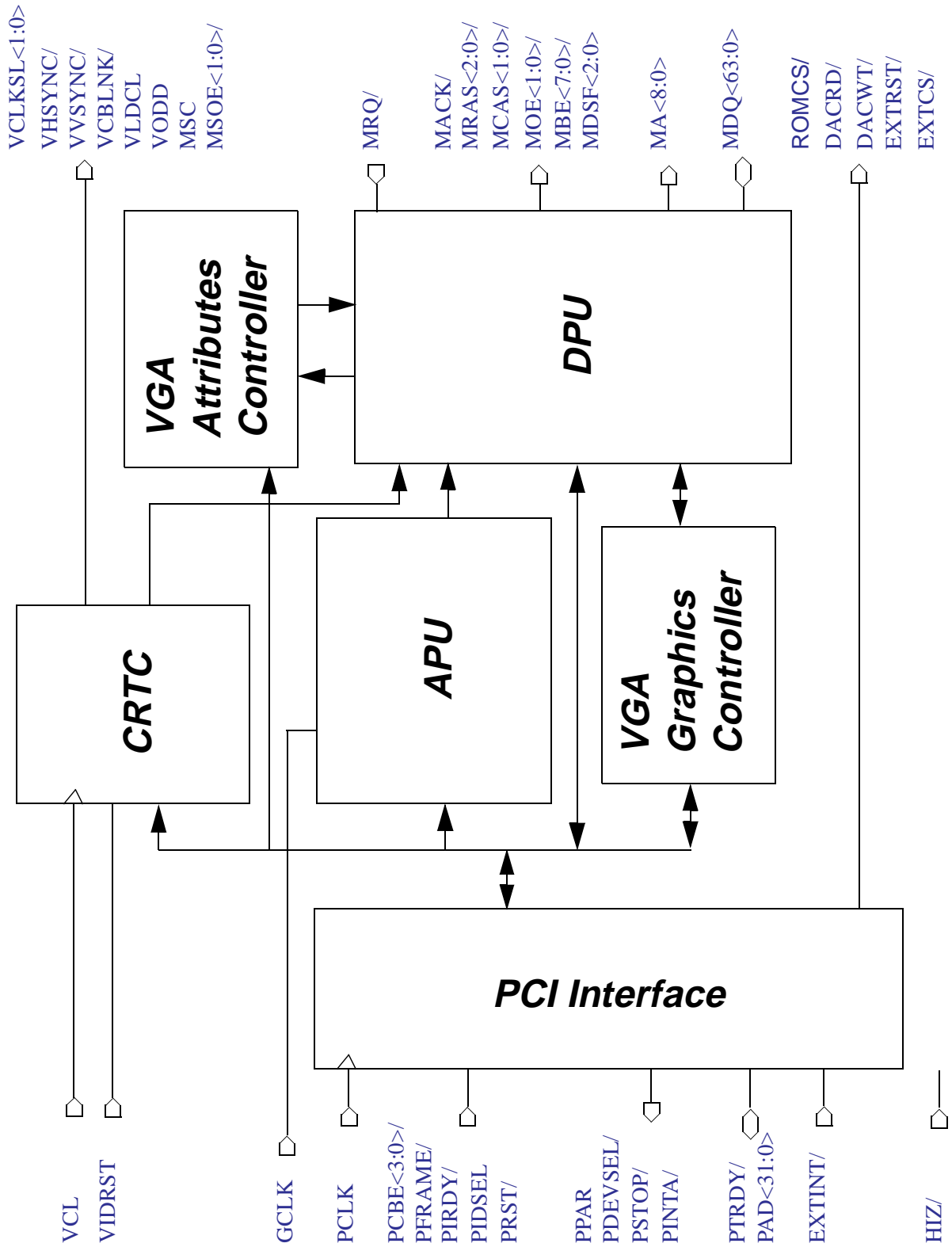
## 2.4 VGA Attributes Controller

This section implements the display refresh for standard VGA modes as well as for all character modes. For these modes, the RAMDAC is driven by the MGA-2064W chip instead of by the serial port of the WRAM.

## 2.5 CRTC

This section generates the horizontal and vertical timing for driving display data and addresses from the frame buffer. The CRTC is VGA-compatible, with some extensions for the Power Graphic modes.

Figure 2-1: MGA-2064W Block Diagram



## 2.6 Address Processing Unit (APU)

This section of the MGA-2064W chip generates the sequencing for drawing operations. Each drawing operation is broken down into a series of read and write commands which are forwarded to the DPU. The APU section includes:

- Generation of the sequences for each drawing operation
- Generation of the addresses
- Processing of the slope for vectors and trapezoid edges
- Rectangle clipping

## 2.7 Data Processing Unit (DPU)

This section manipulates the data according to the currently-selected operation. The DPU also converts read and write commands from the APU into memory cycles to the frame buffer. The DPU includes the:

- Interface to the WRAM bus (VD bus)
- Generation of memory cycles
- Funnel shifter for data alignment
- Boolean ALU
- Patterning circuitry
- Color space converter
- Dithering circuitry
- Data FIFO for bitblt operations
- Color expansion circuitry for character drawing
- Gouraud shading generator
- Depth generation and comparison circuitry
- All control circuitry for external devices

## 2.8 Frame Buffer

The MGA-2064W chip can interface directly with the WRAM chips, using two configurations: interleave and non-interleave.

- The non-interleave configuration supports a 2 or 4 MB frame buffer
- The interleave configuration supports a 4 or 8 MB frame buffer

This allows the MGA-2064W to support 8, 16, 24, and 32 bit/pixel formats and resolutions up to 1920 x 1024. Since WRAM has two ports, the serial port of the WRAM is used for the screen refresh while the random port is devoted to drawing operations. Useful WRAM functions such as split data transfer, block mode, and write/bit are all exploited.





## ***Chapter 3: Resource Mapping***

*This chapter includes:*

Memory Mapping .....	3-2
Configuration Space Mapping .....	3-2
Memory Space Mapping .....	3-3
MGA General Map .....	3-3
MGA Control Aperture Detailed Map .....	3-3
Register Map .....	3-4
I/O Space Mapping .....	3-8
VGA Register Index Summary.....	3-10

## 3.1 Memory Mapping

Note that all addresses and bits within dwords are labelled for a little endian processor (X86 series, for example).

### 3.1.1 Configuration Space Mapping

*Table 3-1: MGA-2064W Configuration Space Mapping*

<i>Address</i>	<i>Name/Note</i>
00-03	<b>DEVID</b>
04-07	<b>DEVCTRL</b>
08-0B	<b>CLASS</b>
0C-0F	<b>HEADER</b>
10-13	<b>MGABASE1</b>
14-17	<b>MGABASE2</b>
18-2F	Reserved <sup>(1)</sup>
30-33	<b>ROMBASE</b>
34-3B	Reserved <sup>(1)</sup>
3C-3F	<b>INTCTRL</b>
40-43	<b>OPTION</b>
44-47	<b>MGA_INDEX</b>
48-4B	<b>MGA_DATA</b>
4C-FF	Reserved <sup>(1)</sup>

<sup>(1)</sup> Writing a reserved location has no effect. Reading a reserved location will give 0's. Access to a reserved location will be decoded.

## 3.2 Memory Space Mapping

### 3.2.1 MGA General Map

Table 3-2: MGA General Map

Address	Condition	Name/Notes
000A0000h-000BFFFFh	<b>GCTL6</b> <3:2> = 00, <b>MISC</b> <1> = 1	VGA frame buffer <sup>(1)</sup> <sup>(2)</sup>
000A0000h-000AFFFFh	<b>GCTL6</b> <3:2> = 01, <b>MISC</b> <1> = 1	
000B0000h-000B7FFFh	<b>GCTL6</b> <3:2> = 10, <b>MISC</b> <1> = 1	
000B8000h-000BFFFFh	<b>GCTL6</b> <3:2> = 11, <b>MISC</b> <1> = 1	
<b>rombase</b> + 0000h to <b>rombase</b> + FFFFh	<b>biosen</b> = 1 (see <b>OPTION</b> ) and <b>romen</b> = 1 (see <b>ROMBASE</b> )	BIOS EPROM <sup>(1)</sup>
<b>mgabase1</b> + 0000h to <b>mgabase1</b> + 3FFFh	MGA control aperture (see Table 3-3)	(1)
<b>mgabase2</b> + 000000h to <b>mgabase2</b> + 7FFFFFFh	Direct frame buffer access aperture	(1)(2)(3)

<sup>(1)</sup> Memory space accesses are decoded only if **memspace** = 1 (see the **DEVCTRL** configuration register).

<sup>(2)</sup> Hardware swapping for big endian support is performed in accordance with the settings of the **OPMODE** register's **dirDataSiz** bits.

<sup>(3)</sup> The usable range depends on the frame buffer configuration. Reading or writing outside the usable range will yield unpredictable results.

### 3.2.2 MGA Control Aperture Detailed Map

Table 3-3: MGA Control Aperture Detailed Map

<b>mgabase1</b> +	Attr.	Mnemonic	Device name
0000h-1BFFh	W	DMAWIN (ILOAD)	7K Pseudo-DMA window <sup>(1)</sup>
	R	DMAWIN (IDUMP)	7K Pseudo-DMA window <sup>(1)</sup>
1C00h-1DFFh	W	DWGREG	Drawing registers <sup>(2)</sup>
1E00h-1EFFh	R/W	HSTREG	Host registers <sup>(2)</sup>
1F00h-1FFFh	R/W	VGAREG	VGA registers <sup>(3)</sup>
2000h-3BFFh		-----	Reserved <sup>(4)</sup>
3C00h-3C1Fh	R/W	RAMDAC	RAMDAC <sup>(5)</sup>
3C20h-3DFFh		-----	Reserved <sup>(4)</sup>
3E00h-3FFFh	R/W	EXPDEV	Expansion <sup>(6)</sup>

<sup>(1)</sup> Hardware swapping for big endian support is performed in accordance with the settings of the **OPMODE** register's **dmaDataSiz** bits.

<sup>(2)</sup> Hardware swapping for big endian support is performed when the **OPTION** configuration register's **powerpc** bit is '1'.

<sup>(3)</sup> VGA registers have been memory mapped to provide access to the **CRTC** registers in order to program MGA video modes when the VGA I/O space is not enabled.

<sup>(4)</sup> Reserved locations are decoded.

<sup>(5)</sup> The exact mapping within this range depends on the RAMDAC and the external connection.

<sup>(6)</sup> The exact mapping within this range depends on the external connections and on the external devices used.

### 3.2.3 Register Map

Table 3-4: Register Mapping (Part 1 of 3)

Byte Address (mgabase1 +)	Access	Name/Notes
1C00	WO <sup>(1)</sup>	<b>DWGCTL</b>
1C04	”	<b>MACCESS</b>
1C08	”	Reserved (MCTLWTST)
1C0C	”	<b>ZORG</b>
1C10	”	<b>PAT0</b>
1C14	”	<b>PAT1</b>
1C18	”	Reserved
1C1C	”	<b>PLNWT</b>
1C20	”	<b>BCOL</b>
1C24	”	<b>FCOL</b>
1C28	”	Reserved
1C2C	”	Reserved (SRCBLT)
1C30	”	<b>SRC0</b>
1C34	”	<b>SRC1</b>
1C38	”	<b>SRC2</b>
1C3C	”	<b>SRC3</b>
1C40	”	<b>XYSTRT</b> <sup>(2)</sup>
1C44	”	<b>XYEND</b> <sup>(2)</sup>
1C48	”	Reserved
1C4C	”	Reserved
1C50	”	<b>SHIFT</b> <sup>(2)</sup>
1C54	”	Reserved
1C58	”	<b>SGN</b> <sup>(2)</sup>
1C5C	”	<b>LEN</b> <sup>(2)</sup>
1C60	”	<b>AR0</b> <sup>(2)</sup>
1C64	”	<b>AR1</b> <sup>(2)</sup>
1C68	”	<b>AR2</b> <sup>(2)</sup>
1C6C	”	<b>AR3</b> <sup>(2)</sup>
1C70	”	<b>AR4</b> <sup>(2)</sup>
1C74	”	<b>AR5</b> <sup>(2)</sup>
1C78	”	<b>AR6</b> <sup>(2)</sup>
1C7C	”	Reserved
1C80	”	<b>CXBNDRY</b> <sup>(2)</sup>
1C84	”	<b>FXBNDRY</b> <sup>(2)</sup>
1C88	”	<b>YDSTLEN</b> <sup>(2)</sup>
1C8C	”	<b>PITCH</b> <sup>(2)</sup>
1C90	”	<b>YDST</b> <sup>(2)</sup>
1C94	”	<b>YDSTORG</b> <sup>(2)</sup>
1C98	”	<b>YTOP</b> <sup>(2)</sup>
1C9C	”	<b>YBOT</b> <sup>(2)</sup>
1CA0	WO <sup>(1)</sup>	<b>CXLEFT</b> <sup>(2)</sup>
1CA4	”	<b>CXRIGHT</b> <sup>(2)</sup>

Table 3-4: Register Mapping (Part 2 of 3)


Byte Address (mgabase1 +)	Access	Name/Notes
1CA8	”	<b>FXLEFT</b> <sup>(2)</sup>
1CAC	”	<b>FXRIGHT</b> <sup>(2)</sup>
1CB0	”	<b>XDST</b> <sup>(2)</sup>
1CB4	”	Reserved
1CB8	”	Reserved
1CBC	”	Reserved
1CC0	”	<b>DR0</b>
1CC4	”	Reserved (DR1)
1CC8	”	<b>DR2</b>
1CCC	”	<b>DR3</b>
1CD0	”	<b>DR4</b>
1CD4	”	Reserved (DR5)
1CD8	”	<b>DR6</b>
1CDC	”	<b>DR7</b>
1CE0	”	<b>DR8</b>
1CE4	”	Reserved (DR9)
1CE8	”	<b>DR10</b>
1CEC	”	<b>DR11</b>
1CF0	”	<b>DR12</b>
1CF4	”	Reserved (DR13)
1CF8	”	<b>DR14</b>
1CFC	”	<b>DR15</b>
1D00-1DFF	”	Same register mapping as the 1C00-1CFC range <sup>(3)</sup>
1E00-1E0F		Reserved
1E10-1E13	RO	<b>FIFOSTATUS</b>
1E14-1E17	RO	<b>Status</b>
1E18-1E1B	WO	<b>ICLEAR</b>
1E1C-1E1F	R/W	<b>IEN</b>
1E20-1E23	RO	<b>VCOUNT</b>
1E24-1E2F		Reserved
1E30-1E33	R/W	Reserved
1E34-1E37	R/W	Reserved
1E38-1E3B	R/W	Reserved
1E3C-1E3F	R/W	Reserved
1E40-1E43	R/W	<b>Reset</b>
1E44-1E53		Reserved
1E54-1E57	R/W	<b>OPMODE</b>
1E58-1E7F		Reserved
1E80-1EBF	W	Reserved
1EE0-1EFF		Reserved
1F00 - 1FBF		Reserved

Table 3-4: Register Mapping (Part 3 of 3)

Byte Address (mgabase1 +)	Access	Name/Notes
1FC0	R/W	Attribute register index
	W	Attribute register
1FC1	R	Attribute register
	W	Reserved
1FC2	W	Miscellaneous output register
	R	Input status register 0
1FC3	R/W	Reserved
1FC4	R/W	Sequencer register index
1FC5	R/W	Sequencer register
1FC6	R/W	Reserved
1FC7	R	RAMDAC state register
	W	Reserved
1FC8	R/W	”
1FC9	R/W	”
1FCA	R	Feature control register
	W	Reserved
1FCB		Reserved
1FCC	R	Miscellaneous output register
	W	Reserved
1FCD		Reserved
1FCE	R/W	Graphic controller register index
1FCF	R/W	Graphic controller register
1FD0		Reserved
1FD1		Reserved
1FD2		”
1FD3		”
1FD4	R/W	CRTC register index
1FD5	R/W	CRTC register
1FD6		Reserved
1FD7		”
1FD8		”
1FD9		”
1FDA	R	Input status register 1
1FDA	W	Feature control register
1FDB		Reserved
1FDC		”
1FDD		”
1FDE	R/W	CRTC extension register index
1FDF	R/W	CRTC extension register
1FC0 - 1FFF		Reserved

<sup>(1)</sup> The drawing registers (1C00 to 1DFF) are all write only. Reading will give zeros.

- (2) Since the address processor can become idle before the data processor, we recommended that you initialize these registers first, in order to take advantage of this idle time.
- (3) When a register is accessed in this range, this instructs the drawing engine to start a drawing cycle.

 **Note:** For the values in [Table 3-4](#), reserved locations should not be accessed. Writing to reserved locations may affect other registers. Reading from reserved locations will return unknown data.

### 3.2.4 I/O Space Mapping

Table 3-5: I/O Space Mapping (Part 1 of 2)

Address <sup>(1)</sup>	Access	Name/Notes
3B0		-- NOT decoded --
3B1		-- NOT decoded --
3B2		-- NOT decoded
3B3		-- NOT decoded --
3B4	R/W	CRTC register index <sup>(2)</sup>
3B5	R/W	CRTC register <sup>(2)</sup>
3B6		-- NOT decoded -- <sup>(2)(3)</sup>
3B7		-- NOT decoded -- <sup>(2)(3)</sup>
3B8		-- NOT decoded --
3B9		-- NOT decoded --
3BA	R	Input status register 1 <sup>(2)</sup>
	W	Feature control register <sup>(2)</sup>
3BB		-- NOT decoded -- <sup>(2)(3)</sup>
3BC		-- NOT decoded --
3BD		-- NOT decoded --
3BE		-- NOT decoded --
3BF		-- NOT decoded --
3C0	R/W	Attribute register index
	W	Attribute register
3C1	R	Attribute register
	W	Reserved -- decoded --
3C2	W	Miscellaneous output register
	R	Input status register 0
3C3		-- NOT decoded -- <sup>(3)</sup>
3C4	R/W	Sequencer register index
3C5	R/W	Sequencer register
3C6	R/W	RAMDAC Display mask register
3C7	R	RAMDAC state register
	W	RAMDAC address register for LUT read
3C8	R/W	RAMDAC address register for LUT write
3C9	R/W	RAMDAC pixel data register
3CA	R	Feature control register
	W	Reserved -- decoded --
3CB		Reserved -- NOT decoded -- <sup>(3)</sup>
3CC	R	Miscellaneous output register
	W	Reserved -- decoded --
3CD		Reserved -- NOT decoded -- <sup>(3)</sup>
3CE	R/W	Graphic controller register index
3CF	R/W	Graphic controller register
3D0		-- NOT decoded --
3D1		-- NOT decoded --



Table 3-5: I/O Space Mapping (Part 2 of 2)

Address <sup>(1)</sup>	Access	Name/Notes
3D2		-- NOT decoded --
3D3		-- NOT decoded --
3D4	R/W	CRTC register index <sup>(4)</sup>
3D5	R/W	CRTC register <sup>(4)</sup>
3D6		-- NOT decoded -- <sup>(3)(4)</sup>
3D7		-- NOT decoded -- <sup>(3)(4)</sup>
3D8		-- NOT decoded --
3D9		-- NOT decoded --
3DA	R	Input status register 1 <sup>(4)</sup>
3DA	W	Feature control register <sup>(4)</sup>
3DB		-- NOT decoded -- <sup>(3)(4)</sup>
3DC		-- NOT decoded --
3DD		-- NOT decoded --
3DE	R/W	CRTC extension register index
3DF	R/W	CRTC extension register

- <sup>(1)</sup> I/O space accesses are decoded only if VGA emulation is active (see the **OPTION** configuration register) and **iospace** = 1 (see the **DEVCTRL** configuration register).
- <sup>(2)</sup> VGA **MISC**<0> register (**ioaddsel** field) is '0' (monochrome emulation).
- <sup>(3)</sup> Word or dword accesses to these specific reserved locations will be decoded. (The PCI convention states that I/O space should only be accessed in bytes, and that a bridge will not perform byte packing.)
- <sup>(4)</sup> **MISC**<0> (**ioaddsel**) is '1' (color emulation).

### 3.3 VGA Register Index Summary

Table 3-6: VGA Indices (Part 1 of 2)

<i>Register Name</i>	<i>Mnemonic</i>	<i>Index</i>
Sequencer registers:	<b>SEQ</b>	<b>seqx</b>
Sequencer Register Index	<b>SEQX</b>	--
Reset	<b>SEQ0</b>	00h
Clocking Mode	<b>SEQ1</b>	01h
Map Mask	<b>SEQ2</b>	02h
Character Map Select	<b>SEQ3</b>	03h
Memory Mode	<b>SEQ4</b>	04h
Reserved	---	05h - 07h
CRTC registers:	<b>CRTC</b>	<b>crtcx</b>
CRTC register index	<b>CRTCX</b>	--
Horizontal Total	<b>CRTC0</b>	00h
Horizontal Display Enable End	<b>CRTC1</b>	01h
Start Horizontal Blanking	<b>CRTC2</b>	02h
End Horizontal Blanking	<b>CRTC3</b>	03h
Start Horizontal Retrace Pulse	<b>CRTC4</b>	04h
End Horizontal Retrace	<b>CRTC5</b>	05h
Vertical Total	<b>CRTC6</b>	06h
Overflow	<b>CRTC7</b>	07h
Preset Row Scan	<b>CRTC8</b>	08h
Maximum Scan Line	<b>CRTC9</b>	09h
Cursor Start	<b>CRTCA</b>	0Ah
Cursor End	<b>CRTCB</b>	0Bh
Start Address High	<b>CRTCC</b>	0Ch
Start Address Low	<b>CRTCD</b>	0Dh
Cursor Location High	<b>CRTCE</b>	0Eh
Cursor Location Low	<b>CRTCF</b>	0Fh
Vertical Retrace Start	<b>CRTC10</b>	10h
Vertical Retrace End	<b>CRTC11</b>	11h
Vertical Display Enable End	<b>CRTC12</b>	12h
Offset	<b>CRTC13</b>	13h
Underline Location	<b>CRTC14</b>	14h
Start Vertical Blank	<b>CRTC15</b>	15h
End Vertical Blank	<b>CRTC16</b>	16h
CRTC Mode Control	<b>CRTC17</b>	17h
Line Compare	<b>CRTC18</b>	18h
Reserved -- read as 0	----	19h - 21h
CPU Read Latch	<b>CRTC22</b>	22h
Reserved -- read as 0	----	23h
Attributes address/data select	<b>CRTC24</b>	24h
Reserved -- read as 0	----	25h
Attributes address	<b>CRTC26</b>	26h
Reserved -- read as 0	----	27h

Table 3-6: VGA Indices (Part 2 of 2)

<b>Register Name</b>	<b>Mnemonic</b>	<b>Index</b>
Reserved -- read as 0	----	28h - 3Fh
Attribute Controller registers:	<b>ATTR</b>	<b>attrx</b>
Attribute Controller Index	<b>ATTRX</b>	--
Palette entry 0	<b>ATTR0</b>	00h
Palette entry 1	<b>ATTR1</b>	01h
Palette entry 2	<b>ATTR2</b>	02h
Palette entry 3	<b>ATTR3</b>	03h
Palette entry 4	<b>ATTR4</b>	04h
Palette entry 5	<b>ATTR5</b>	05h
Palette entry 6	<b>ATTR6</b>	06h
Palette entry 7	<b>ATTR7</b>	07h
Palette entry 8	<b>ATTR8</b>	08h
Palette entry 9	<b>ATTR9</b>	09h
Palette entry A	<b>ATTRA</b>	0Ah
Palette entry B	<b>ATTRB</b>	0Bh
Palette entry C	<b>ATTRC</b>	0Ch
Palette entry D	<b>ATTRD</b>	0Dh
Palette entry E	<b>ATTRE</b>	0Eh
Palette entry F	<b>ATTRF</b>	0Fh
Attribute Mode Control	<b>ATTR10</b>	10h
Overscan Color	<b>ATTR11</b>	11h
Color Plane Enable	<b>ATTR12</b>	12h
Horizontal Pel Panning	<b>ATTR13</b>	13h
Color Select	<b>ATTR14</b>	14h
Reserved	----	15h - 1Fh
Graphic Controller registers:	<b>GCTL</b>	<b>gctlx</b>
Graphic Controller Index	<b>GCTLX</b>	--
Set/Reset	<b>GCTL0</b>	00h
Enable Set/Reset	<b>GCTL1</b>	01h
Color Compare	<b>GCTL2</b>	02h
Data Rotate	<b>GCTL3</b>	03h
Read Map Select	<b>GCTL4</b>	04h
Graphic Mode	<b>GCTL5</b>	05h
Miscellaneous	<b>GCTL6</b>	06h
Color Don't Care	<b>GCTL7</b>	07h
Bit Mask	<b>GCTL8</b>	08h
Reserved	---	09h - 0Fh
General registers		
Miscellaneous Output register	<b>MISC</b>	
Feature Control register	<b>FEAT</b>	
Input Status register 0	<b>INSTS0-</b>	
Input Status register 1	<b>INSTS1</b>	
DAC Status	<b>DACSTAT</b>	





## ***Chapter 4: Register Descriptions***

*This chapter includes:*

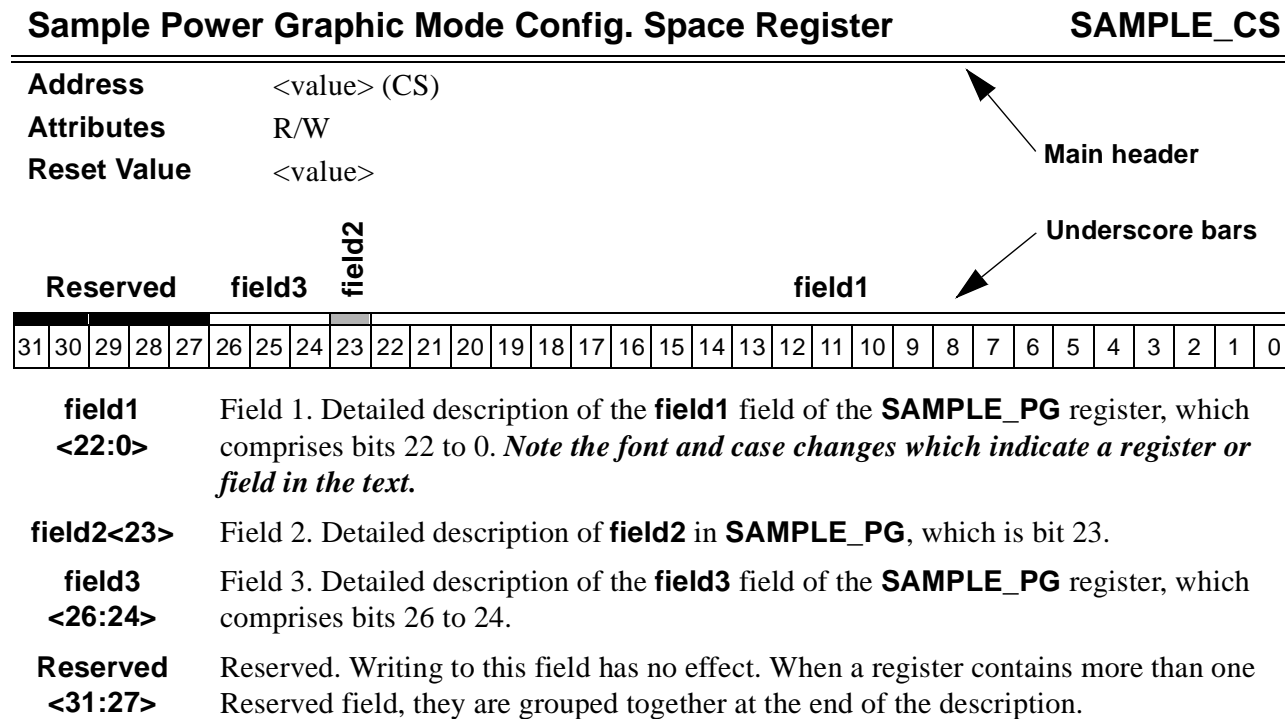
Power Graphic Mode Register Descriptions.....	4-2
Power Graphic Mode	
Configuration Space Registers .....	4-2
Power Graphic Mode	
Memory Space Registers .....	4-16
VGA Mode Register Descriptions .....	4-79

Note: The registers in this chapter are arranged in alphabetical order within each section.

## 4.1 Power Graphic Mode Register Descriptions

### 4.1.1 Power Graphic Mode Configuration Space Registers

Power Graphic mode register descriptions contain a (double-underlined) main header which indicates the register's mnemonic abbreviation and full name. Below the main header, the configuration space address (30h, for example), attributes, and reset value for the register are provided. Next, an illustration identifies the bit fields, which are then described in detail underneath. The reserved fields are underscored by black bars, and all other fields are delimited by alternating white and gray bars.



#### Memory Address

The addresses of all the Power Graphic mode registers are provided in [Chapter 3](#). Note: CS indicates that the address lies within the configuration space.

#### Attributes

The Power Graphic mode configuration space register attributes are:

- RO: There are no writable bits.
- R/W: The state of the written bits can be read.
- BYTE: 8-bit access to the register is possible.
- WORD: 16-bit access to the register is possible.
- DWORD: 32-bit access to the register is possible.
- STATIC: The contents of the register will not change during an operation.

#### Reset Value

Most bits are reset on hard reset. Some bits are also reset on soft reset, and they are underlined when they appear in the register description headers.

- 000X 0000 h (h = Hexadecimal, X = undefined)
- S = bit's reset value is affected by a strap setting.

**Class Code****CLASS**

<b>Address</b>	08h (CS)
<b>Attributes</b>	RO, BYTE/WORD/DWORD, STATIC
<b>Reset Value</b>	0000 0011 S000 0000 0000 0000 0000 0000 b

class																revision															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**revision** <7:0> Holds the current chip revision (01h).

**class** <31:8> Identifies the generic function of the device and a specific register-level programming interface as per the PCI specification. Two values can be read in this field according to the vga boot strap, which is sampled on hard reset.

<i>vgaboot strap</i>	<i>Value</i>	<i>Meaning</i>
0	038000h	Non-Super VGA display controller
1	030000h	Super VGA compatible controller

The sampled state of the vga boot strap (pin MDQ<5>) can be read through this register.





**Device Control****... DEVCTRL**


---



---

<b>devsel</b> RO <26:25>	Device select timing. Specifies the timing of devsel. It is read as '01'.
<b>sigtargab</b> RO <27>	MGA never signals a target abort. This bit read as '0' and writing has no effect.
<b>sigsyserr</b> RO <30>	MGA does not assert SERR/. Writing has no effect. Reading will give 0's.
<b>detparerr</b> RO <31>	MGA does not detect parity errors. Writing has no effect. Reading will give 0's.
<b>Reserved:</b>	<4> <22:9> <24> <29:28>
	Reserved. Writing to these fields has no effect. Reading will give 0's.

**DEVID**

**Device Identification**

**Address**            00h (CS)  
**Attributes**        RO, BYTE/WORD/DWORD, STATIC  
**Reset Value**       0000 0101 0001 1001 0001 0000 0010 1011 b

**device**

**vendor**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

**vendor**            This field contains the Matrox manufacturer identifier for PCI: 102Bh.  
**<15:0>**

**device**            This field contains the Matrox device identifier for this product: 0519h.  
**<31:16>**

**Header****HEADER**


---

**Address**            0Ch (CS)  
**Attributes**        RO, BYTE/WORD/DWORD, STATIC  
**Reset Value**        0000 0000 0000 0000 0000 0000 0000 0000 b

<b>Reserved</b>								<b>header</b>								<b>Reserved</b>															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**header**            This field specifies the layout of bytes 10h through 3Fh in the configuration space and  
**<23:16>**           also indicates that the current device is a single function device. This field is read as 00h.

**Reserved:**        **<15:0>** **<31:24>**

Reserved. Writing to these fields has no effect. Reading will give 0's.

**INTCTRL****Interrupt Control**

**Address** 3Ch (CS)  
**Attributes** R/W, BYTE/WORD/DWORD, STATIC  
**Reset Value** 0000 0000 0000 0000 0000 0001 1111 1111 b

**Reserved****intpin****intline**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

**intline**  
**R/W <7:0>** Interrupt line routing. The field is read/writable and reset to FFh upon hard reset. It is up to the configuration program to determine which interrupt level is tied to the MGA interrupt line and program the **intline** field accordingly (Note: the value 'FF' indicates either 'unknown' or 'no connection').

**intpin**  
**RO <15:8>** Selected interrupt pins. Read as 1h to indicate that one PCI interrupt line is used (PCI specifies that if there is one interrupt line, it must be connected to the [PINTA/](#) signal).

**Reserved**  
**<31:16>** Reserved. Writing to these fields has no effect. Reading will give 0's.

**MGA Indirect Access Data****MGA\_DATA**

<b>Address</b>	48h (CS)
<b>Attributes</b>	R/W, BYTE/WORD/DWORD, STATIC
<b>Reset Value</b>	None

**data**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

**data**  
**<31:0>**

Data. Will read or write data at the control register address provided by **MGA\_INDEX**.

Note: It is possible that an access to the **data** field will not be responded to when the **memspace** field of **DEVCTRL** is set to 0.

The **MGA\_INDEX** and **MGA\_DATA** registers cannot be used in Pseudo DMA mode (see [page 5-16](#)).

**MGA\_INDEX****MGA Indirect Access Index**

<b>Address</b>	44h (CS)
<b>Attributes</b>	R/W, BYTE/WORD/DWORD, STATIC
<b>Reset Value</b>	0000 0000 0000 0000 0000 0000 0000 0000 b

**Reserved****index**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

**index**  
**<13:2>** Dword index. Used to reach any of the registers that are mapped into the MGA control aperture through the configuration space. This mechanism should be used for initialization purposes only, since it is inefficient. This ‘back door’ access to the control register can be useful when the control aperture cannot be mapped below the 1 Mbyte limit of the real mode of an x86 processor (during BIOS execution, for example).

**Reserved**  
**<31:14>** Reserved. Writing to this field has no effect. Reading will give 0’s.

**Reserved**  
**<1:0>** Reserved. Writing to this field has no effect. Reading will return unreliable results.

The **MGA\_INDEX** and **MGA\_DATA** registers cannot be used in Pseudo DMA mode (see [page 5-16](#)).

**MGA Control Aperture Base Address****MGABASE1**

<b>Address</b>	10h (CS)
<b>Attributes</b>	R/W, BYTE/WORD/DWORD, STATIC
<b>Reset Value</b>	0000 0000 0000 0000 0000 0000 0000 0000 b

mgabase1														Reserved							prefetchable	type	memspaceind								
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**memspace-ind RO <0>** The hard coded '0' indicates that the map is in the memory space.

**type RO <2:1>** The hard coded '00' instructs the configuration program to locate the aperture anywhere within the 32-bit address space.

**prefetchable RO <3>** The hard coded '0' indicates that this space cannot be prefetchable.

**Reserved <13:4>** Reserved. Writing to this field has no effect. Reading will give 0's.

**mgabase1 <31:14>** Specifies the base address of the MGA memory mapped control registers (16 Kbyte control aperture).

In situations where the MGA control aperture overlaps the MGA frame buffer aperture and/or the ROM aperture, the following precedence order will be used, listed from highest to lowest:

1. BIOS EPROM (highest precedence)
2. MGA control aperture
3. VGA frame buffer aperture
4. MGA frame buffer aperture (lowest precedence)

**MGABASE2****MGA frame buffer aperture address**

<b>Address</b>	14h (CS)
<b>Attributes</b>	R/W, BYTE/WORD/DWORD, STATIC
<b>Reset Value</b>	0000 0000 0000 0000 0000 0000 0000 1000 b

mgabase2										Reserved										prefetchable	type	memspaceind									
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**memspace-ind**  
**RO <0>**

The hard coded '0' indicates that the map is in the memory space.

**type**  
**RO <2:1>**

The hard coded '00' instructs the configuration program to locate the aperture anywhere within the 32-bit address space.

**prefetchable**  
**RO <3>**

The hard coded '1' indicates that this space can be prefetchable (better system performance can be achieved when the bridge enables prefetching into that range).

**Reserved**  
**<22:4>**

Reserved. Writing to this field has no effect. Reading will give 0's.

**mgabase2**  
**<31:23>**

Specifies the PCI start address of the 8 Mbytes of MGA memory space in the PCI map. In situations where the MGA control aperture overlaps the MGA frame buffer aperture and/or the ROM aperture, the following precedence order will be used, listed from highest to lowest:

1. BIOS EPROM (highest precedence)
2. MGA control aperture
3. VGA frame buffer aperture
4. MGA frame buffer aperture (lowest precedence)

When **mgamode** = 0 (**CRTCEXT3**<7>), the full frame buffer aperture is not available.





## ... OPTION

## Option

	<p>During the reset period, the refresh request is continuously forced to its inactive state so that no RAM activity will occur. By maintaining the reset low for 200 us, a proper RAM initialization will occur (valid for power up or after an invalid RAM operation).</p> <ul style="list-style-type: none"> <li>• Note that when <b>rfhcnt</b> = 0, the minimum frequency is 4 Mhz (when <b>nogscale</b> = 1), and 16 Mhz (when <b>nogscale</b> = 0).</li> </ul>
<b>eeepromwt</b> <b>&lt;20&gt;</b>	EEPROM write enable. When set to 1, a write access to the BIOS EPROM aperture will program that location. When set to 0, write access to the BIOS EPROM aperture has no effect.
<b>nogscale</b> <b>&lt;21&gt;</b>	Graphic clock pre-scaler. When set to 0, the gclk signal is divided by 4 internally, and when set to 1, gclk is not divided. The gclk divider could be used when the PLL is not able to lower the gclk enough to achieve power-down mode.
<b>productid</b> <b>RO &lt;28:24&gt;</b>	Product ID bits. Sampled state of the MDQ<4:0> pin after a hard reset.  These bits are available to help board designers encode their product options so that the software and diagnostics can know which options are installed. (This field could encode the amount of memory and the RAMDAC type, if a writable ROM is present, and so on). These bits do not control hardware within the chip.
<b>noretry&lt;29&gt;</b>	Retry disable. This field disables retries by the MGA-2064W under certain specific conditions, in order to address a problem with a certain PCI chipset.  <ul style="list-style-type: none"> <li>• 0: This bit should be set to '0', unless you experience problems with retries.</li> <li>• 1: When set to '1', retries are disabled during the initial data phase of any transfer to the MGA-2064W. Set this bit to '1' if you experience problems with retries.</li> </ul>
<b>biosen</b> <b>R/W&lt;30&gt;</b>	BIOS enable.  <ul style="list-style-type: none"> <li>• 0: The <b>ROMBASE</b> space is automatically disabled.</li> <li>• 1: The <b>ROMBASE</b> space is enabled - <b>rombase</b> must be correctly initialized since it contains unpredictable data.</li> </ul>
<b>powerpc</b> <b>&lt;31&gt;</b>	Power PC mode.  <ul style="list-style-type: none"> <li>• 0: No special swapping is performed. The host processor is assumed to be of little endian type.</li> <li>• 1: Enables byte swapping for the memory range <b>mgabase1</b> + 1C00 to <b>mgabase1</b> + 1EFF. This swapping allows a big endian processor to access the information in the same manner as a little endian processor.</li> </ul>
<b>Reserved:</b>	<b>&lt;23:22&gt; &lt;15:13&gt; &lt;11:9&gt; &lt;7:0&gt;</b>  Reserved. Writing to these fields has no effect. Reading will give 0's.

## ROM Base Address

## ROMBASE

<b>Address</b>	30h (CS)
<b>Attributes</b>	R/W, BYTE/WORD/DWORD, STATIC
<b>Reset Value</b>	0000 0000 0000 0000 0000 0000 0000 0000 b

rombase																Reserved															romen	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

**romen**  
<0> ROM enable. This field can assume different attributes, depending on the contents of the **biosen** field. This allows booting with or without the BIOS EPROM (typically, a motherboard implementation will boot the MGA without the BIOS, while an add-on adapter will boot the MGA with the BIOS EPROM).

<b>biosen</b>	<b>romen</b> attribute
0	RO (read as 0)
1	R/W

**Reserved**  
<15:1> Reserved. Writing to this field has no effect. Reading will give 0's.

**rombase**  
<31:16> ROM base address. Specifies the base address of the EPROM. This field can assume different attributes, depending on the contents of **biosen**.

<b>biosen</b>	<b>rombase</b> attribute
0	RO (read as 0)
1	R/W

Note: the exact size of the EPROM used is application-specific (could be 32K or 64K).

In situations where the MGA control aperture overlaps the MGA frame buffer aperture and/or the ROM aperture, the following precedence order will be used, listed from highest to lowest:

1. BIOS EPROM (highest precedence)
2. MGA control aperture
3. VGA frame buffer aperture
4. MGA frame buffer aperture (lowest precedence)

Even if MGA supports only an 8-bit-wide EPROM, this does not constitute a system performance limitation, since the PCI specification requires the configuration software to move the EPROM contents into shadow memory and execute the code at that location.

The sampled state of the biosen strap (pin MDQ<6>) can be determined by doing a write test to this register.

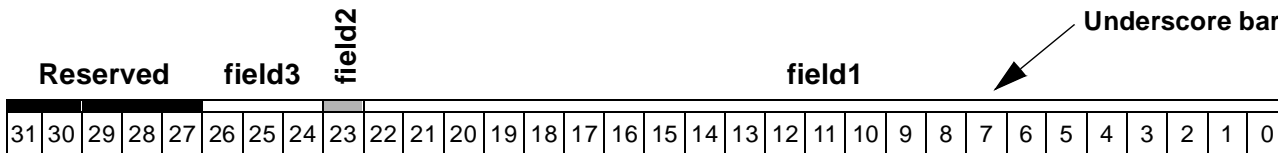
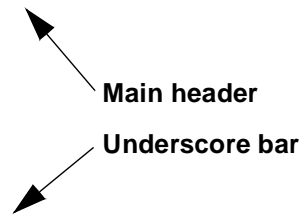
## 4.1.2 Power Graphic Mode Memory Space Registers

Power Graphic mode register descriptions contain a (double-underlined) main header which indicates the register's mnemonic abbreviation and full name. Below the main header, the memory address (1C00h, for example), attributes, and reset value for the register are provided. Next, an illustration identifies the bit fields, which are then described in detail underneath. The reserved fields are underscored by black bars, and all other fields are delimited by alternating white and gray bars.

### Sample Power Graphic Mode Memory Space Register

**SAMPLE\_PG**

**Address** <value>  
**Attributes** R/W  
**Reset Value** <value>



**field1** <22:0> Field 1. Detailed description of the **field1** field of the **SAMPLE\_PG** register, which comprises bits 22 to 0. *Note the font and case changes which indicate a register or field in the text.*

**field2**<23> Field 2. Detailed description of **field2** in **SAMPLE\_PG**, which is bit 23.

**field3** <26:24> Field 3. Detailed description of the **field3** field of the **SAMPLE\_PG** register, which comprises bits 26 to 24.

**Reserved** <31:27> Reserved. Writing to this field has no effect. When a register contains more than one Reserved field, they are grouped together at the end of the description.

### Memory Address

The addresses of all the Power Graphic mode registers are provided in [Chapter 3](#).

MEM: The address lies in the memory space, IO: The address lies in the I/O space.

### Attributes

The Power Graphic mode attributes are:

- RO: There are no writable bits.
- WO: The state of the written bits cannot be read.
- R/W: The state of the written bits can be read.
- BYTE: 8-bit access to the register is possible.
- WORD: 16-bit access to the register is possible.
- DWORD: 32-bit access to the register is possible.
- STATIC: The contents of the register will not change during an operation.
- DYNAMIC: The contents of the register might change during an operation.
- FIFO: Data written to this register will pass through the BFIFO.

### Reset Value

Most bits are reset on hard reset. Some bits are also reset on soft reset, and they are underlined when they appear in the register description headers.

- 000X 0000 h (h = Hexadecimal, X = undefined)
- 0000 0000 0X00 0000 0000 0000 0000 b (b = Binary, 00 = reset on soft and hard reset; see above)
- S = bit's reset value is affected by a strap setting.

**Multi-Purpose Address 0****AR0**

<b>Address</b>	<b>mgabase1</b> + 1C60h (MEM)
<b>Attributes</b>	WO, FIFO, DYNAMIC, DWORD
<b>Reset Value</b>	Unknown

Reserved														ar0																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**Note:** Writing to this register when the **DWGCTL** register's **arzero** bit = 1 will produce unpredictable results. Make sure that a '0' has been written to **arzero** prior to accessing **AR0**.

<b>ar0</b> <b>&lt;17:0&gt;</b>	Address register 0. The ar0 field is an 18-bit signed value in two's complement notation. <ul style="list-style-type: none"> <li>• For AUTOLINE, this register holds the x end address (in pixels). See the <b>XYEND</b> register on page 4-68.</li> <li>• For LINE, it holds 2 x 'b'.</li> <li>• For a filled trapezoid, it holds 'dYI'.</li> <li>• For a BLIT, <b>ar0</b> holds the line end source address (in pixels).</li> <li>• For an ILOAD_SCALE or ILOAD_FILTER, <b>ar0</b> holds the destination end address (in pixels) minus one line.</li> </ul>
<b>Reserved</b> <b>&lt;31:18&gt;</b>	Reserved. Writing to this field has no effect.

**AR1****Multi-Purpose Address 1**

<b>Address</b>	<b>mgabase1</b> + 1C64h (MEM)
<b>Attributes</b>	WO, FIFO, DYNAMIC, DWORD
<b>Reset Value</b>	Unknown

**Reserved****ar1**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

**Note:** Writing to this register when the **DWGCTL** register's **arzero** bit = 1 will produce unpredictable results. Make sure that a '0' has been written to **arzero** prior to accessing **AR1**.

**ar1**  
**<23:0>**

Address register 1. The **ar1** field is a 24-bit signed value in two's complement notation. This register is also loaded when **ar3** is accessed.

- For LINE, it holds the error term (initially  $2 \times 'b' - 'a' - [\text{sd}y]$ ).
- This register does not need to be loaded for AUTOLINE.
- For a filled trapezoid, it holds the error term in two's complement notation; initially:

$$'err1' = [\text{sd}x1] ? 'dX1' + 'dY1' - 1 : -'dX1'$$

- For a BLIT, **ar1** holds the line start source address (in pixels). Because the start source address is also required by **ar3**, and because **ar1** is loaded when writing **ar3** this register doesn't need to be explicitly initialized.
- In the ILOAD\_SCALE and ILOAD\_FILTER algorithms, **ar1** contains the destination starting address (in pixels) minus one line. Because the same value is also required by **ar3** and because **ar1** is loaded when writing **ar3**, this register doesn't need to be explicitly initialized.

**Reserved**  
**<31:24>**

Reserved. Writing to this field has no effect.

**Multi-Purpose Address 2****AR2**

<b>Address</b>	<b>mgabase1</b> + 1C68h (MEM)
<b>Attributes</b>	WO, FIFO, DYNAMIC, DWORD
<b>Reset Value</b>	Unknown

Reserved														ar2																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**Note:** Writing to this register when the **DWGCTL** register's **arzero** bit = 1 will produce unpredictable results. Make sure that a '0' has been written to **arzero** prior to accessing **AR2**.

**ar2**  
**<17:0>**

Address register 2. The **ar2** field is an 18-bit signed value in two's complement notation.

- For AUTOLINE, this register holds the y end address (in pixels). See the **XYEND** register on page 4-68.
- For LINE, it holds the minor axis error increment (initially  $2 \times 'b' - 2 \times 'a'$ ).
- For a filled trapezoid, it holds the minor axis increment ( $-|dXI|$ ).
- For ILOAD\_SCALE, it holds the error increment which is the source dimension for the x axis. ( $dXsrc$ )
- For ILOAD\_FILTER, it holds the error increment which is the source dimension after the filter process for the x axis. ( $2 * dXsrc - 1$ )

This register is not used for BLIT operations.

**Reserved**  
**<31:18>**

Reserved. Writing to this field has no effect.

**AR3****Multi-Purpose Address 3**

<b>Address</b>	<b>mgabase1</b> + 1C6Ch (MEM)
<b>Attributes</b>	WO, FIFO, DYNAMIC, DWORD
<b>Reset Value</b>	Unknown

Reserved						spage		ar3																							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**Note:** Writing to this register when the **DWGCTL** register's **arzero** bit = 1 will produce unpredictable results. Make sure that a '0' has been written to **arzero** prior to accessing **AR3**.

**ar3**  
**<23:0>**

Address register 3. The **ar3** field is a 24-bit signed value in two's complement notation or a 24-bit unsigned value.

- This register is used during AUTOLINE, but does not need to be initialized.
- This register is not used for LINE without auto initialization, nor is it used by TRAP.
- In the two-operand Blit algorithms and ILOAD **ar3** contains the source current address (in pixels). This value must be initialized as the starting address for a Blit. The source current address is always linear.
- In the ILOAD\_SCALE and ILOAD\_FILTER algorithms, **ar3** contains the destination current address (in pixels) minus one line. This value must be initialized as the destination starting address minus one line.

**spage**  
**<25:24>**

These two bits are used as an extension to **ar3** in order to generate a 26-bit source or pattern address (in pixels). They are not modified by ALU operations.

In BLIT operations, the spage field is only used with monochrome source data.

The **spage** field is not used for TRAP, LINE or AUTOLINE operations.

**Reserved**  
**<31:26>**

Reserved. Writing to this field has no effect.



**Multi-Purpose Address 4****AR4**

<b>Address</b>	<b>mgabase1</b> + 1C70h (MEM)
<b>Attributes</b>	WO, FIFO, DYNAMIC, DWORD
<b>Reset Value</b>	Unknown

Reserved														ar4																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**Note:** Writing to this register when the **DWGCTL** register's **arzero** bit = 1 will produce unpredictable results. Make sure that a '0' has been written to **arzero** prior to accessing **AR4**.

**ar4**  
**<17:0>**

Address register 4. The **ar4** field is an 18-bit signed value in two's complement notation.

- For TRAP, it holds the error term. Initially:

$$\text{'errr'} = [\text{sdxr}] ? \text{'dXr'} + \text{'dYr'} - 1 : -\text{'dXr'}$$

- This register is used during AUTOLINE, but doesn't need to be initialized.
- This register is not used for LINE or BLIT operations.
- For the ILOAD\_SCALE and ILOAD\_FILTER, it holds the error term, but it doesn't need to be initialized.

**Reserved**  
**<31:18>**

Reserved. Writing to this field has no effect.

**AR5****Multi-Purpose Address 5**

<b>Address</b>	<b>mgabase1</b> + 1C74h (MEM)
<b>Attributes</b>	WO, FIFO, DYNAMIC, DWORD
<b>Reset Value</b>	Unknown

Reserved														ar5																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**Note:** Writing to this register when the **DWGCTL** register's **arzero** bit = 1 will produce unpredictable results. Make sure that a '0' has been written to **arzero** prior to accessing **AR5**.

**ar5**  
**<17:0>**

Address register 5. The **ar5** field is an 18-bit signed value in two's complement notation.

- At the beginning of AUTOLINE, **ar5** holds the x start address (in pixels). See the **XYSTRT** register on page 4-69. At the end of AUTOLINE the register is loaded with the x end, so it is not necessary to reload the register when drawing a polyline.
- This register is not used for LINE without auto initialization.
- For TRAP, it holds the minor axis increment ( $-|dXr|$ ).
- In BLIT algorithms, **ar5** holds the pitch (in pixels) of the source operand. A negative pitch value specifies that the source is scanned from bottom to top while a positive pitch value specifies a top to bottom scan.

**Reserved**  
**<31:18>**

Reserved. Writing to this field has no effect.

## Multi-Purpose Address 6

AR6

<b>Address</b>	<b>mgabase1</b> + 1C78h (MEM)
<b>Attributes</b>	WO, FIFO, DYNAMIC, DWORD
<b>Reset Value</b>	Unknown

Reserved														ar6																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**Note:** Writing to this register when the **DWGCTL** register's **arzero** bit = 1 will produce unpredictable results. Make sure that a '0' has been written to **arzero** prior to accessing **AR6**.

**ar6**  
<17:0>

Address register 6. This field is an 18-bit signed value in two's complement notation. It is sign extended to 24 bits before being used by the ALU.

- At the beginning of AUTOLINE, **ar6** holds the y start address (in pixels). See the **XYSTRT** register on page 4-69. During AUTOLINE processing, this register is loaded with the signed y displacement. At the end of AUTOLINE the register is loaded with the y end, so it is not necessary to reload the register when drawing a polyline.
- This register is not used for LINE without auto initialization.
- For TRAP, it holds the major axis increment ('dYr').
- For ILOAD\_SCALE, it holds the error increment which is the source dimension (in pixels) minus the destination dimension for the x axis. (dXsrc - dXdst) Note that **ar6** must be less than or equal to zero.
- For ILOAD\_FILTER, it holds the error increment which is the source dimension (in pixels) minus the destination dimension for the x axis. (2 \* dXsrc - 1 - dXdst) Note that **ar6** must be less than or equal to zero.

This register is not used for BLIT or IDUMP operations.

**Reserved**  
<31:18>

Reserved. Writing to this field has no effect.

**BCOL****Background Color**

<b>Address</b>	<b>mgabase1</b> + 1C20h (MEM)
<b>Attributes</b>	WO, FIFO, STATIC, DWORD
<b>Reset Value</b>	Unknown

**backcol**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

**backcol** <31:0> Background color. The **backcol** field is used by the color expansion module to generate the source pixels when the background is selected.

- In 8 and 16 bits/pixel configurations, all bits in **backcol**<31:0> are used, so the color information must be replicated on all bytes.
- In 24 bits/pixel, when not in block mode, **backcol**<31:24> is not used.
- In 24 bits/pixel, when in block mode, all **backcol** bits are used.

Refer to ‘Pixel Format’ on page 5-11 for the the definition of the slice in each mode.

**Clipper X Boundary****CXBNDRY**

**Address**            **mgabase1** + 1C80h (MEM)  
**Attributes**        WO, FIFO, STATIC, DWORD  
**Reset Value**        Unknown

Reserved					cxright											Reserved					cxleft										
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

The **CXBNDRY** register is not a physical register. It is simply an alternate way to load the **CXRIGHT** and **CXLEFT** registers.

**cxleft**  
**<10:0>**            Clipper x left boundary. See the **CXLEFT** register on page 4-26.

**cxright**  
**<26:16>**            Clipper x right boundary. See the **CXRIGHT** register on page 4-27.

**Reserved:**    **<15:11>** **<31:27>**

Reserved. Writing to these fields has no effect.

**CXLEFT****Clipper X Minimum Boundary**

<b>Address</b>	<b>mgabase1</b> + 1CA0h (MEM)
<b>Attributes</b>	WO, FIFO, STATIC, DWORD
<b>Reset Value</b>	Unknown

Reserved															cxleft																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**cxleft**  
**<10:0>** Clipper x left boundary. The **cxleft** field contains an unsigned 11-bit value which is interpreted as a positive pixel address and compared with the current **xdst** (see **XDST** on page 4-67). The value of **xdst** must be greater than or equal to **cxleft** to be inside the drawing window.

Note that since the **cxleft** value is interpreted as positive, any negative **xdst** value is automatically outside the clipping window.

There is no way to disable clipping.

**Reserved**  
**<31:11>** Reserved. Writing to this field has no effect.

**Clipper X Maximum Boundary****CXRIGHT**

<b>Address</b>	<b>mgabase1</b> + 1CA4h (MEM)
<b>Attributes</b>	WO, FIFO, STATIC, DWORD
<b>Reset Value</b>	Unknown

Reserved											cxright																				
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**cxright**  
**<10:0>** Clipper x right boundary. The **cxright** field contains an unsigned 11-bit value which is interpreted as a positive pixel address and compared with the current **xdst** (see **XDST** on page 4-67). The value of **xdst** must be less than or equal to **cxright** to be inside the drawing window.

There is no way to disable clipping.

**Reserved**  
**<31:11>** Reserved. Writing to this field has no effect.

**DR0****Data ALU 0**

<b>Address</b>	<b>mgabase1 + 1CC0h (MEM)</b>
<b>Attributes</b>	WO, FIFO, DYNAMIC, DWORD
<b>Reset Value</b>	Unknown

**dr0**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

**dr0**  
**<31:0>**

Data ALU register 0.

- For TRAP with z, the **DR0** register is used to scan the left edge of the trapezoid and must be initialized with its starting z value. In this case, **DR0** is a signed 17.15 value in two's complement notation.
- For LINE with z, the **DR0** register holds the z value for the current drawn pixel and must be initialized with the starting z value. In this case, **DR0** is a signed 17.15 value in two's complement notation.



**Data ALU 2****DR2**

<b>Address</b>	<b>mgabase1</b> + 1CC8h (MEM)
<b>Attributes</b>	WO, FIFO, STATIC, DWORD
<b>Reset Value</b>	Unknown

**dr2**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

**dr2**  
**<31:0>** Data ALU register 2.

- For TRAP with z, the **DR2** register holds the z increment value along the x axis. In this case, **DR2** is a signed 17.15 value in two's complement notation.
- For LINE with z, the **DR2** register holds the z increment value along the major axis. In this case, **DR2** is a signed 17.15 value in two's complement notation.

**DR3****Data ALU 3**

<b>Address</b>	<b>mgabase1 + 1CCCh (MEM)</b>
<b>Attributes</b>	WO, FIFO, STATIC, DWORD
<b>Reset Value</b>	Unknown

**dr3**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

**dr3**  
**<31:0>**

Data ALU register 3.

- For TRAP with z, **DR3** register holds the z increment value along the y axis. In this case, **DR3** is a signed 17.15 value in two's complement notation.
- For LINE with z, **DR3** register holds the z increment value along the diagonal axis. In this case, **DR3** is a signed 17.15 value in two's complement notation.

**Data ALU 4****DR4**

**Address**            **mgabase1 + 1CD0h (MEM)**  
**Attributes**        **WO, FIFO, DYNAMIC, DWORD**  
**Reset Value**        **Unknown**

Reserved								dr4																							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**dr4**                    Data ALU register 4. This field holds a signed 9.15 value in two's complement notation.  
**<23:0>**

- For TRAP with z, the **DR4** register is used to scan the left edge of the trapezoid for the red color (Gouraud shading). This register must be initialized with its starting red color value.
- For TEXTURE\_TRAP, this register is not used, and will be corrupted.
- For LINE with z, the **DR4** register holds the current red color value for the currently drawn pixel. This register must be initialized with the starting red color.

**Reserved**            Reserved. Writing to this field has no effect.  
**<31:24>**

**DR6****Data ALU 6**

**Address**            **mgabase1 + 1CD8h (MEM)**  
**Attributes**        **WO, FIFO, STATIC, DWORD**  
**Reset Value**        **Unknown**

**Reserved****dr6**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

**dr6**            Data ALU register 6. This field holds a signed 9.15 value in two's complement notation.

**<23:0>**

- For TRAP with z, the **DR6** register holds the red increment value along the x axis.
- For TEXTURE\_TRAP, this register is not used.
- For LINE with z, the **DR6** register holds the red increment value along the major axis.

**Reserved**        Reserved. Writing to this field has no effect.

**<31:24>**

**Data ALU 7****DR7**

**Address**            **mgabase1 + 1CDCh (MEM)**  
**Attributes**        **WO, FIFO, STATIC, DWORD**  
**Reset Value**        **Unknown**

**Reserved****dr7**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

**dr7**            Data ALU register 7. This field holds a signed 9.15 value in two's complement notation.  
**<23:0>**

- For TRAP with z, the **DR7** register holds the red increment value along the y axis.
- For TEXTURE\_TRAP, this register is not used.
- For LINE with z, the **DR7** register holds the red increment value along the diagonal axis.

**Reserved**        Reserved. Writing to this field has no effect.  
**<31:24>**

**DR8****Data ALU 8**

**Address**            **mgabase1 + 1CE0h (MEM)**  
**Attributes**        **WO, FIFO, DYNAMIC, DWORD**  
**Reset Value**        **Unknown**

**Reserved****dr8**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

**dr8**  
**<23:0>**

Data ALU register 8. This field holds a signed 9.15 value in two's complement notation.

- For TRAP with z, the **DR8** register is used to scan the left edge of the trapezoid for the green color (Gouraud shading). This register must be initialized with its starting green color value.
- For TEXTURE\_TRAP, this register is not used, but will be corrupted.
- For LINE with z, the **DR8** register holds the current green color value for the currently drawn pixel. This register must be initialized with the starting green color.

**Reserved**  
**<31:24>**

Reserved. Writing to this field has no effect.

**Data ALU 10****DR10**

**Address**            **mgabase1 + 1CE8h (MEM)**  
**Attributes**        **WO, FIFO, STATIC, DWORD**  
**Reset Value**        **Unknown**

**Reserved****dr10**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

**dr10**  
**<23:0>**

Data ALU register 10. This field holds a signed 9.15 value in two's complement notation.

- For TRAP with z, the **DR10** register holds the green increment value along the x axis.
- For TEXTURE\_TRAP, this register is not used.
- For LINE with z, the **DR10** register holds the green increment value along the major axis.

**Reserved**  
**<31:24>**

Reserved. Writing to this field has no effect.

**DR11****Data ALU 11**

**Address**            **mgabase1 + 1CECh (MEM)**  
**Attributes**        **WO, FIFO, STATIC, DWORD**  
**Reset Value**        **Unknown**

**Reserved****dr11**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

**dr11**  
**<23:0>**

Data ALU register 11. This field holds a signed 9.15 value in two's complement notation.

- For TRAP with z, the **DR11** register holds the green increment value along the y axis.
- For TEXTURE\_TRAP, this register is not used.
- For LINE with z, the **DR11** register holds the green increment value along the diagonal axis.

**Reserved**  
**<31:24>**

Reserved. Writing to this field has no effect.



**Data ALU 12****DR12**

<b>Address</b>	<b>mgabase1</b> + 1CF0h (MEM)
<b>Attributes</b>	WO, FIFO, DYNAMIC, DWORD
<b>Reset Value</b>	Unknown

Reserved								dr12																							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**dr12**  
**<23:0>** Data ALU register 12. This field holds a signed 9.15 value in two's complement notation.

- For TRAP with z, the **DR12** register is used to scan the left edge of the trapezoid for the blue color (Gouraud shading). This register must be initialized with its starting blue color value.
- For TEXTURE\_TRAP, this register is not used, but will be corrupted.
- For LINE with z, the **DR12** register holds the blue color value for the currently drawn pixel. This register must be initialized with the starting blue color.

**Reserved**  
**<31:24>** Reserved. Writing to this field has no effect.

**DR14****Data ALU 14**

**Address**            **mgabase1 + 1CF8h (MEM)**  
**Attributes**        **WO, FIFO, STATIC, DWORD**  
**Reset Value**        **Unknown**

**Reserved****dr14**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

**dr14**  
**<23:0>**

Data ALU register 14. This field holds a signed 9.15 value in two's complement notation.

- For TRAP with z, the **DR14** register holds the blue increment value along the x axis.
- For TEXTURE\_TRAP, this register is not used.
- For LINE with z, the **DR14** register holds the blue increment value along the major axis.

**Reserved**  
**<31:24>**

Reserved. Writing to this field has no effect.

**Data ALU 15****DR15**

**Address**            **mgabase1** + 1CFCh (MEM)  
**Attributes**        WO, FIFO, STATIC, DWORD  
**Reset Value**        Unknown

**Reserved****dr15**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

**dr15**  
**<23:0>**

Data ALU register 15. This field holds a signed 9.15 value in two's complement notation.

- For TRAP with z, the **DR15** register holds the blue increment value along the y axis.
- For TEXTURE\_TRAP, this register is not used.
- For LINE with z, the **DR15** register holds the blue increment value along the diagonal axis.

**Reserved**  
**<31:24>**

Reserved. Writing to this field has no effect.

## DWGCTL

## Drawing Control

**Address** mgabase1 + 1C00h (MEM)  
**Attributes** WO, FIFO, STATIC, DWORD  
**Reset Value** 0000 0000 0000 0000 0000 0000 0000 0000 b

Reserved	transc	pattern	bltmod		Reserved	trans		bop		Reserved	shftzero	sgnzero	arzero	solid	zmode		linear	atype	opcode												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**opcode** <3:0> Operation code. The **opcode** field defines the operation that is selected by the drawing engine. This field also affects the operation of the WRAM interface section.

		<i>opcode</i>	
<i>Function</i>	<i>Sub-Function</i>	<i>Value</i>	<i>Mnemonic</i>
Lines		0000	LINE_OPEN
	AUTO	0001	AUTOLINE_OPEN
	WRITE LAST	0010	LINE_CLOSE
	AUTO, WRITE LAST	0011	AUTOLINE_CLOSE
Trapezoid		0100	TRAP
	texture mapping	0101	TEXTURE_TRAP
Bitblt	WRAM -> WRAM	1000	BITBLT
	WRAM -> WRAM	1100	FBITBLIT
	HOST -> WRAM	1001	ILOAD
	HOST -> WRAM	1101	ILOAD_SCALE
	HOST -> WRAM	1111	ILOAD_FILTER
	WRAM -> HOST	1010	IDUMP
	Reserved	0110	
	”	0111	
	”	1011	
	”	1110	

## Drawing Control

... DWGCTL ...

**atype** <6:4> Access type. The **atype** field is used to define the type of access performed to the WRAM.

<b>atype</b>		<i>WRAM Access</i>
<i>Value</i>	<i>Mnemonic</i>	
000	RPL	Write (replace)
001	RSTR	Read-modify-write (raster)
010		Reserved
011	ZI	Depth mode with Gouraud
100	BLK	Block write mode (1)
101		Reserved
110		Reserved
111	I	Gouraud (with depth compare)

(1) When block mode is selected, only RPL operations can be performed. Even if the **bop** field is programmed to a different value, RPL will be used.

**linear** <7> Linear mode. Specifies whether the bitblt is linear or xy.

- 0: xy bitblt
- 1: linear bitblt

**zmode** <10:8> The z drawing mode. This field must be valid for drawing using depth. This field specifies the type of comparison to use.

<b>zmode</b>		<i>Pixel Update</i>
<i>Value</i>	<i>Mnemonic</i>	
000	NOZCMP	Always
001		Reserved
010	ZE	When depth is =
011	ZNE	When depth is < >
100	ZLT	When depth is <
101	ZLTE	When depth is <=
110	ZGT	When depth is >
111	ZGTE	When depth is >=

## ... DWGCTL ...

## Drawing Control

**solid** <11> Solid line or constant trapezoid. The solid register is not a physical register. It provides an alternate way to load the **SRC** registers (see [page 4-64](#)).

- 0: No effect
- 1: **SRC0** <= FFFFFFFFh  
**SRC1** <= FFFFFFFFh  
**SRC2** <= FFFFFFFFh  
**SRC3** <= FFFFFFFFh

Setting solid is useful for line drawing with no linestyle, or for trapezoid drawing with no patterning. It forces the color expansion circuitry to provide the foreground color during a line or a trapezoid drawing.

**arzero** <12> **AR** register at zero. The **arzero** field provides an alternate way to set certain **AR** registers (see descriptions starting on [page 4-17](#)).

- 0: No effect
- 1: **AR0** <= 0h  
**AR1** <= 0h  
**AR2** <= 0h  
**AR4** <= 0h  
**AR5** <= 0h  
**AR6** <= 0h

Setting **arzero** is useful when drawing rectangles, and also for certain blit operations.

In the case of rectangles (TRAP **opcod**):

```
dYl <= 0 (AR0)
errl <= 0 (AR1)
-|dXl| <= 0 (AR2)
errr <= 0 (AR4)
-|dXr| <= 0 (AR5)
dYr <= 0 (AR6)
```

Writing to the **ARx** registers when **arzero** = 1 will produce unpredictable results. Make sure that a '0' has been written to **arzero** prior to accessing the **ARx** registers.

**sgnzero** <13> Sign register at zero. The **sgnzero** bit provides an alternate way to set all the fields in the **SGN** register.

- 0: No effect
- 1: **SGN** <= 0h

Setting **sgnzero** is useful during TRAP and some Blit operations.

For TRAP: **scanleft** = 0Horizontal scan right  
**sdxl** = 0Left edge in increment mode  
**sdxr** = 0Right edge in increment mode  
**sdyl** = 0iy (see **PITCH** on [page 4-59](#)) is added to  
**ydst** (see **YDST** on [page 4-71](#))

## Drawing Control

... DWGCTL ...

For BLIT: **scanleft** = 0Horizontal scan right  
**sdxl** = 0Left edge in increment mode  
**sdxr** = 0Right edge in increment mode)  
**sdyl** = 0iy is added to **ydst**

Writing to the **SGN** register when **sgnzero** = 1 will produce unpredictable results. Make sure that a '0' has been written to **sgnzero** prior to accessing the **SGN** register.

**shftzero**  
<14>

Shift register at zero. The **shftzero** bit provides an alternate way to set all the fields of the **SHIFT** register.

- 0: No effect
- 1: **SHIFT** <= 0h

Writing to the **SHIFT** register when **shftzero** = 1 will produce unpredictable results. Make sure that a '0' has been written to **shftzero** prior to accessing the **SHIFT** register.

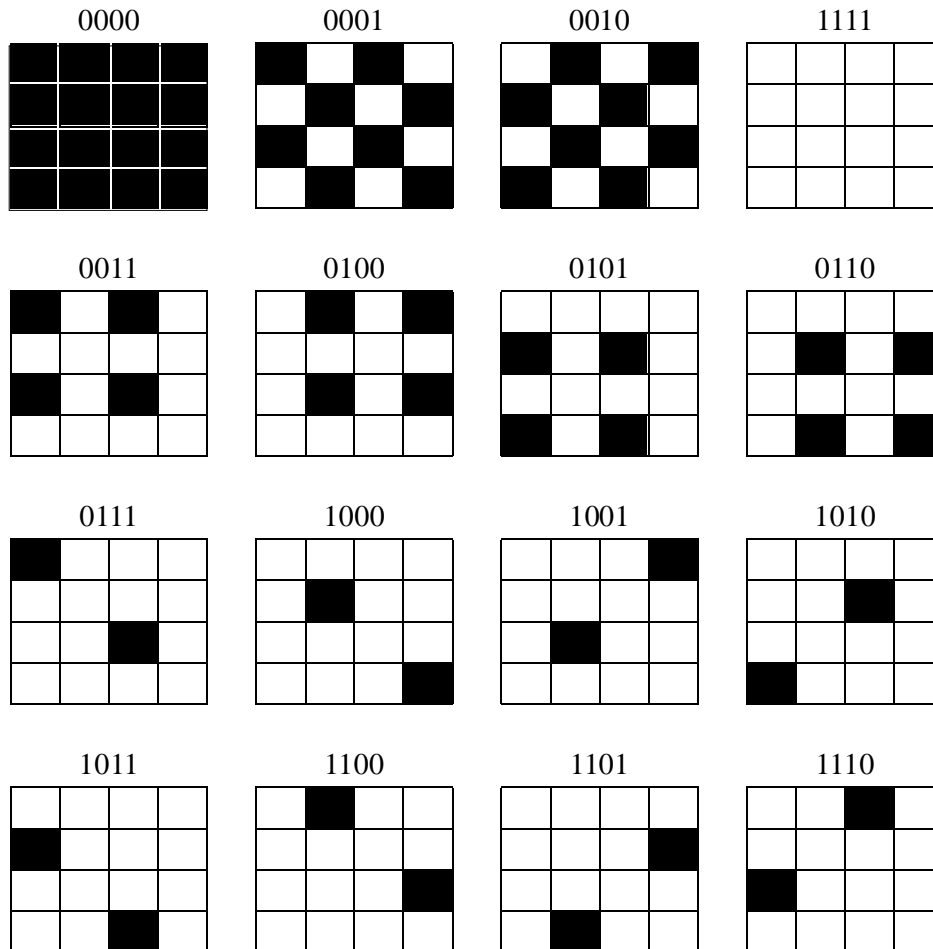
**bop**  
<19:16>

Boolean operation between a source and a destination slice. The table below shows the various functions performed by the Boolean ALU for 8, 16, 24, and 32 bits/pixel. During block mode operations, **bop** must be set to Ch.

<b>bop</b>	<i>Function</i>
0000	0
0001	~(D   S)
0010	D & ~S
0011	~S
0100	(~D) & S
0101	~D
0110	D ^ S
0111	~(D & S)
1000	D & S
1001	~(D ^ S)
1010	D
1011	D   ~S
1100	S
1101	(~D)  S
1110	D   S
1111	1

**trans**  
**<23:20>**

Translucidity. Specify the percentage of opaqueness of the object. The opaqueness is realized by writing one of 'n' pixels. The **trans** field specifies the following transparency pattern (where black squares are opaque and white squares are transparent):





## Drawing Control

## ... DWGCTL

**bltmod** Blit mode selection. This field is defined as used during BLIT and ILOAD operations.  
**<28:25>**

bltmod		
Value	Mnemonic	Usage
0000	BMONOLEF	Source operand is monochrome in 1 bpp. For ILOAD, the source data is in little endian format.
0100	BMONOWF	Source operand is monochrome in 1 bpp. For ILOAD, the source data is in Windows format.
0001	BPLAN	Source operand is monochrome from one plane.
0010	BFCOL	Source operand is color. Source is formatted when it comes from host.
1110	BUYUV	Source operand is color. For ILOAD, the source data is in 4:2:2 YUV format.
0011	BU32BGR	Source operand is color. For ILOAD, the source data is in 32 bpp, BGR format.
0111	BU32RGB	Source operand is color. For ILOAD, the source data is in 32 bpp, RGB format.
1011	BU24BGR	Source operand is color. For ILOAD, the source data is in 24 bpp, BGR format.
1111	BU24RGB	Source operand is color. For ILOAD, the source data is in 24 bpp, RGB format.
0101		Reserved
0110		”
1000		”
1001		”
1010		”
1100		”
1101		”

- For line drawing with line style, this field must have the value BFCOL in order to handle the line style properly.
- For a WRAM-to-WRAM BITBLT operation, hardware fast clipping will be enabled if BFCOL is specified.
- The field is also used for the IDUMP and TEXTURE\_TRAP operations.

Refer to the subsections contained in ‘Drawing in Power Graphic Mode’ on page 5-16 for more information on how to use this field. That section also presents the definition of the various pixel formats.

## ... DWGCTL

## Drawing Control

---

---

<b>pattern</b> <b>&lt;29&gt;</b>	Patterning enable. This bit specifies if the patterning is enabled when performing BLIT operations. <ul style="list-style-type: none"><li>• 0: Patterning is disabled.</li><li>• 1: Patterning is enabled.</li></ul>
<b>transc</b> <b>&lt;30&gt;</b>	Transparency color enabled. This field must be valid for color expansion blits and for vectors that have a line style. This bit specifies if the background color is used. <ul style="list-style-type: none"><li>• 0: Background color is opaque.</li><li>• 1: Background color is transparent.</li></ul>
<b>Reserved:</b>	<b>&lt;15&gt; &lt;24&gt; &lt;31&gt;</b> Reserved. Writing to these fields has no effect.

---

**Foreground Color**
**FCOL**


---

<b>Address</b>	<b>mgabase1</b> + 1C24h (MEM)
<b>Attributes</b>	WO, FIFO, STATIC, DWORD
<b>Reset Value</b>	Unknown

**forc**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

**forc**  
**<31:0>**      Foreground color. The **forc** field is used by the color expansion module to generate the source pixels when the foreground is selected.

- In 8 and 16 bits/pixel configurations, all bits in **forc**<31:0> are used, so the color information must be replicated on all bytes.
- In 24 bits/pixel, when not in block mode, **forc**<31:24> is not used.
- In 24 bits/pixel, when in block mode, all **forc** bits are used.

Refer to ‘Pixel Format’ on page 5-11 for the the definition of the slice in each mode.

Part of the **forc** register is also used for Gouraud shading to generate the alpha bits. In 32 bpp, bits 31 to 24 originate from **forc**<31:24>. In 16 bpp, when 5:5:5 mode is selected, bits 16 originates from **forc**<31>.

## FIFOSTATUS

## Bus FIFO Status

<b>Address</b>	<b>mgabase1</b> + 1E10 (MEM)
<b>Attributes</b>	RO, DYNAMIC, BYTE/WORD/DWORD
<b>Reset Value</b>	0000 0000 0000 0000 0000 00 <u>10</u> 00 <u>10</u> 0000 b

Reserved																bempty	bfull	Reserved	fifocount												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- fifocount** <5:0> Indicates the number of free locations in the Bus FIFO. On soft or hard reset, the contents of the Bus FIFO are flushed and the FIFO count is set to 32.
- bfull** <8> Bus FIFO full flag. When set to '1', indicates that the Bus FIFO is full.
- bempty** <9> Bus FIFO empty flag. When set to '1', indicates that the Bus FIFO is empty. This bit is identical to **fifocount**<5>.
- Reserved:** <7:6> <31:10>

Reserved. Writing to these fields has no effect. Reading will give 0's.

There is no need to poll the **bfull** or **fifocount** values before writing to the BFIFO: circuitry in the MGA watches the BFIFO level and generates target retries until a free location becomes available, or until a retry limit has been exceeded (in which case, it might indicate an abnormal engine lock-up).

Even if the machine that reads the Bus FIFO is asynchronous with the PCI interface, a sample and hold circuit has been added to provide a correct, non-changing value during the full PCI read cycle (the **fifocount** value, **bfull**, and **bempty** flag states are sampled at the start of the PCI access).

**X Address (Boundary)****FXBNDRY**

<b>Address</b>	<b>mgabase1</b> + 1C84h (MEM)
<b>Attributes</b>	WO, FIFO, DYNAMIC, DWORD
<b>Reset Value</b>	Unknown

<b>fxright</b>																<b>fxleft</b>															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

The **FXBNDRY** register is not a physical register. It is simply an alternate way to load the **FXRIGHT** and **FXLEFT** registers.

- fxleft**  
**<15:0>** Filled object x left coordinate. Refer to the **FXLEFT** register for a detailed description.
- fxright**  
**<31:16>** Filled object x right coordinate. See the **FXRIGHT** register on page 4-51.

**FXLEFT****X Address (Left)**

<b>Address</b>	<b>mgabase1</b> + 1CA8h (MEM)
<b>Attributes</b>	WO, FIFO, DYNAMIC, DWORD
<b>Reset Value</b>	Unknown

**Reserved****fxleft**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

**fxleft**  
**<15:0>**

Filled object x left coordinate. The **fxleft** field contains the x coordinate (in pixels) of the left boundary of any filled object being drawn. It is a 16-bit signed value in two's complement notation.

- The **fxleft** field is not used for line drawing.
- During filled trapezoid drawing, **fxleft** is updated during the left edge scan.
- During a BLIT operation, **fxleft** is static, and specifies the left pixel boundary of the area being written to.

**Reserved**  
**<31:16>**

Reserved. Writing to this field has no effect.

**X Address (Right)****FXRIGHT**

<b>Address</b>	<b>mgabase1</b> + 1CACH (MEM)
<b>Attributes</b>	WO, FIFO, DYNAMIC, DWORD
<b>Reset Value</b>	Unknown

Reserved																fxright															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**fxright**  
**<15:0>** Filled object x right coordinate. The **fxright** field contains the x coordinate (in pixels) of the right boundary of any filled object being drawn. It is a 16-bit signed value in two's complement notation.

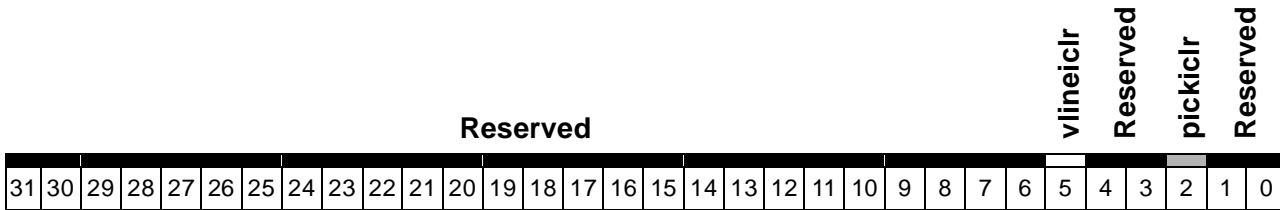
- The **fxright** field is not used for line drawing.
- During filled trapezoid drawing, **fxright** is updated during the right edge scan.
- During a BLIT operation, **fxright** is static, and specifies the right pixel boundary of the area being written to.

**Reserved**  
**<31:16>** Reserved. Writing to this field has no effect.

**ICLEAR**

**Interrupt Clear**

**Address**            **mgabase1 + 1E18 (MEM)**  
**Attributes**        **WO, DYNAMIC, BYTE/WORD/DWORD**  
**Reset Value**        **0000 0000 0000 0000 0000 0000 0000 0000 b**



**pickiclr**            Pick interrupt clear. When a ‘1’ is written to this bit, the pick interrupt pending flag is cleared.  
**<2>**

**vlineiclr**            Vertical line interrupt clear. When a ‘1’ is written to this bit, the vertical line interrupt pending flag is cleared.  
**<5>**

**Reserved:**    **<1:0> <4:3> <31:6>**

Reserved. Writing to these fields has no effect. Reading will give 0’s.



## Interrupt Enable

IEN

<b>Address</b>	<b>mgabase1 + 1E1C (MEM)</b>
<b>Attributes</b>	R/W, STATIC, BYTE/WORD/DWORD
<b>Reset Value</b>	0000 0000 0000 0000 0000 0000 <u>0000</u> <u>0000</u> b

Reserved																				extien	vlineien	Reserved	pickien	Reserved									
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
<b>pickien</b>		Picking interrupt enable. When set to '1', enables interrupts if a picking interrupt occurs.																															
<b>&lt;2&gt;</b>																																	
<b>vlineien</b>					Vertical line interrupt enable. When set to '1', an interrupt will be generated when the vertical line counter equals the vertical line interrupt count.																												
<b>&lt;5&gt;</b>																																	
<b>extien</b>						External interrupt enable. When set to '1', an external interrupt will contribute to the generation of a PCI interrupt on the <a href="#">PINTA/</a> line.																											
<b>&lt;6&gt;</b>																																	
<b>Reserved:</b>																																	
<b>&lt;1:0&gt; &lt;4:3&gt; &lt;31:7&gt;</b>																																	
Reserved. Writing to these fields has no effect. Reading will give 0's.																																	

**LEN****Length**

**Address**            **mgabase1 + 1C5Ch (MEM)**  
**Attributes**        **WO, FIFO, DYNAMIC, DWORD**  
**Reset Value**        **Unknown**

**Reserved****length**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

**length**            Length. The length bit is a 16-bit unsigned value.

**<15:0>**

- The **length** field does not require initialization for auto-init vectors.
- For a vector draw, **length** is programmed with the number of pixels to be drawn.
- For Blits and trapezoid fills, **length** is programmed with the number of lines to be filled or BLITed.

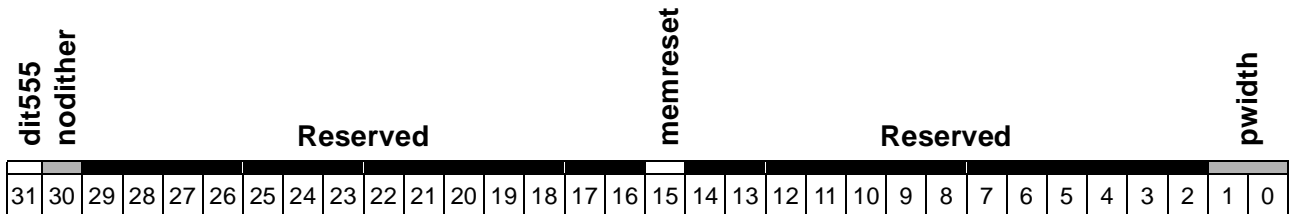
**Reserved**        Reserved. Writing to this field has no effect.

**<31:16>**

## Memory Access

## MACCESS

<b>Address</b>	<b>mgabase1</b> + 1C04h (MEM)
<b>Attributes</b>	WO, FIFO, STATIC, DWORD
<b>Reset Value</b>	0000 0000 0000 0000 0000 0000 0000 0000 b



**pwidth** <1:0> Pixel width. Specifies the normal pixel width for drawing.

<b>pwidth</b>		
<i>Value</i>	<i>Mnemonic</i>	<i>Mode</i>
00	PW8	8 bpp
01	PW16	16 bpp
10	PW32	32 bpp
11	PW24	24 bpp

**memreset** <15> Resets the WRAM. When this bit is set to '1', the memory sequencer will generate one reset cycle to the WRAM. Refer to Section 5.3.2 on page 5-14 for instructions on when to use this field.

**nodither** <30> Enable/disable dithering.

- 0: Dithering is performed on unformatted ILOAD, ZI, and I trapezoids.
- 1: Dithering is disabled.

**dit555** <31> Dither 5:5:5 mode. This field should normally be set to 0, except for 16 bit/pixel configurations, when it affects dithering and shading.

- 0: The pixel format is 5:6:5
- 1: The pixel format is 5:5:5

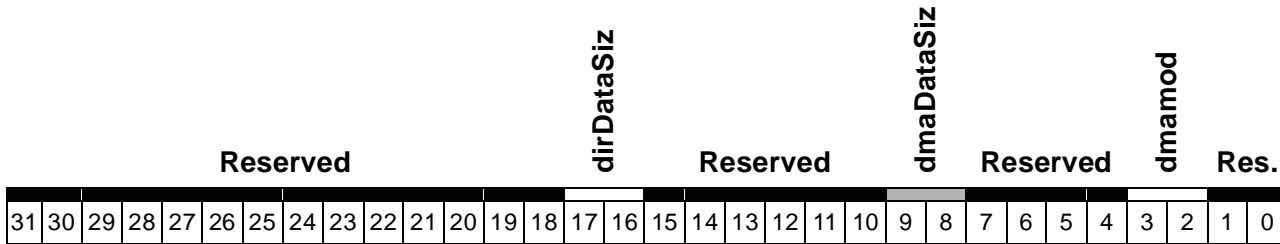
**Reserved** <29:16> <14:2>

Reserved. Writing to this field has no effect.

**OPMODE**

**Operating Mode**

**Address** mgabase1 + 1E54 (MEM)  
**Attributes** R/W, STATIC BYTE/WORD/DWORD  
**Reset Value** 0000 0000 0000 0000 0000 0000 0000 0000 b



**dmamod** <3:2> Select the Pseudo DMA transfer mode.

dmamod<1:0>	DMA Transfer Mode Description
00	DMA General Purpose Write
01	DMA BLIT Write
10	DMA Vector Write
11	Reserved

**dmaDataSiz** <9:8> DMAWIN data size. Controls a hardware swapper for big endian processor support during access to the DMAWIN space. Normally, **dmaDataSiz** is '00' for any DMA mode except DMA BLIT WRITE.

dmaDatSiz <1:0>	Endian Format	Data Size	Internal Data Written to Register			
			reg<31:24>	reg<23:16>	reg<15:8>	reg<7:0>
00	little	any	PAD<31:24>	PAD<23:16>	PAD<15:8>	PAD<7:0>
	big	8 bpp				
01	big	16 bpp	PAD<23:16>	PAD<31:24>	PAD<7:0>	PAD<15:8>
10	big	32 bpp	PAD<7:0>	PAD<15:8>	PAD<23:16>	PAD<31:24>
11	big	Reserved				

**dirDataSiz** <17:16> Direct frame buffer access data size. Controls a hardware swapper for big endian processor support during access to the full frame buffer aperture or the VGA frame buffer aperture.

dirDatSiz <1:0>	Endian Format	Data Size	Internal Data Written to Register			
			mem<31:24>	mem<23:16>	mem<15:8>	mem<7:0>
00	little	any	PAD<31:24>	PAD<23:16>	PAD<15:8>	PAD<7:0>
	big	8 bpp				
01	big	16 bpp	PAD<23:16>	PAD<31:24>	PAD<7:0>	PAD<15:8>
10	big	32 bpp	PAD<7:0>	PAD<15:8>	PAD<23:16>	PAD<31:24>
11	big	Reserved				

---

**Operating Mode...****OPMODE**

---

**Reserved:** <1:0> <7:4> <15:10> <31:18>

Reserved. Writing to these fields has no effect. Reading will give 0's.

Writing to byte 0 of this register will terminate the current DMA sequence and initialize the machine for the new mode (even if the value did not change). This effect should be used to break an incomplete packet.



## Memory Pitch

## PITCH

<b>Address</b>	<b>mgabase1</b> + 1C8Ch (MEM)
<b>Attributes</b>	WO, FIFO, STATIC, DWORD
<b>Reset Value</b>	Unknown

<b>Reserved</b>																<b>ylin</b>	<b>Res.</b>				<b>iy</b>										
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**iy**  
**<11:0>**

The y increment. This field is a 12-bit unsigned value. The y increment value is a pixel unit, which must be a multiple of 32 (the five LSB = 0) and must be less than or equal to 2048. The **iy** field specifies the increment to be added to or subtracted from **ydst** (see [YDST on page 4-71](#)) between two destination lines. The **iy** field is also used as the multiplier factor for linearizing the **ydst** register.

Note that only a few values are supported for linearization. If the pitch selected can't be linearized, the **ylin** bit should be used to disable the linearization operation. The following table provides the supported pitches for linearization:

<i>Pitch</i>	<b>iy</b>
640	001010000000
768	001100000000
800	001100100000
960	001111000000
1024	010000000000
1152	010010000000
1280	010100000000
1600	011001000000
1920	011110000000
2048	100000000000

This register must be loaded with a multiple of values according to the table below due to a restriction involving block mode (see '[Constant Shaded Trapezoids / Rectangle Fills](#)' on page 5-23) and fast blits (see '[Two-operand Fast Bitblts](#)' on page 5-30). Refer also to [page 4-41](#) for additional restrictions that apply to block mode (**atype** = BLK).

<b>pwidth</b>	<i>Non-Interleave</i>	<i>Interleave</i>
PW8	64	128
PW16	32	64
PW24	64	128
PW32	32	32

**ylin**  
**<15>**

The y linearization. This bit specifies whether the address must be linearized or not.

- 0: The address is an xy address, so it must be linearized by the hardware
- 1: The address is already linear

**Reserved:** **<31:16>** **<14:12>**

Reserved. Writing to these fields has no effect.

**PLNWT****Plane Write Mask**

<b>Address</b>	<b>mgabase1</b> + 1C1Ch (MEM)
<b>Attributes</b>	WO, FIFO, STATIC, DWORD
<b>Reset Value</b>	Unknown

**plnwrmsk**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

**plnwrmsk <31:0>** Plane write mask. Plane(s) to be protected during any write operations. During any intensity buffer write operations, the contents of this register are transmitted to the WRAMs through the MDQ<63:0> bus, where they are latched on the falling edge of RAS/. The plane write mask is not used for z cycles, or for direct write access (all planes are written in this case).

- 0 = inhibit write
- 1 = permit write

The bits from **plnwrmsk** are output on the MDQ<31:0> signal and also on MDQ<63:32>. In 8 and 16 bit/pixel configurations, all bits in **plnwrmsk** are used, so the mask information must be replicated on all bytes. In 24 bits/pixel, the plane masking feature is limited to the case of all three colors having the same mask. The four bytes of **plnwrmsk** must be identical.

Refer to 'Pixel Format' on page 5-11 for the the definition of the slice in each mode.



## Reset

## RST

<b>Address</b>	<b>mgabase1</b> + 1E40 (MEM)
<b>Attributes</b>	R/W, STATIC, BYTE/WORD/DWORD
<b>Reset Value</b>	0000 0000 0000 0000 0000 0000 0000 0000 b

## Reserved

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

softreset

**softreset**  
**<0>**

Soft reset. When set to '1', resets all software-resettable bits. This has the effect of flushing all FIFOs, invalidating the direct access read cache, aborting the current drawing instruction, and terminating the current memory cycle. A soft reset will not generate invalid memory cycles, and memory contents are preserved as long as the soft reset bit is not maintained to '1' for more than 2 mS. The **softreset** signal takes place at the end of the PCI write cycle. The reset bit must be maintained to '1' for a minimum of 10 uS to ensure correct reset. After that period, a '0' must be programmed to remove the soft reset.

Refer to [Section 5.3.2 on page 5-14](#) for instructions on when to use this field.

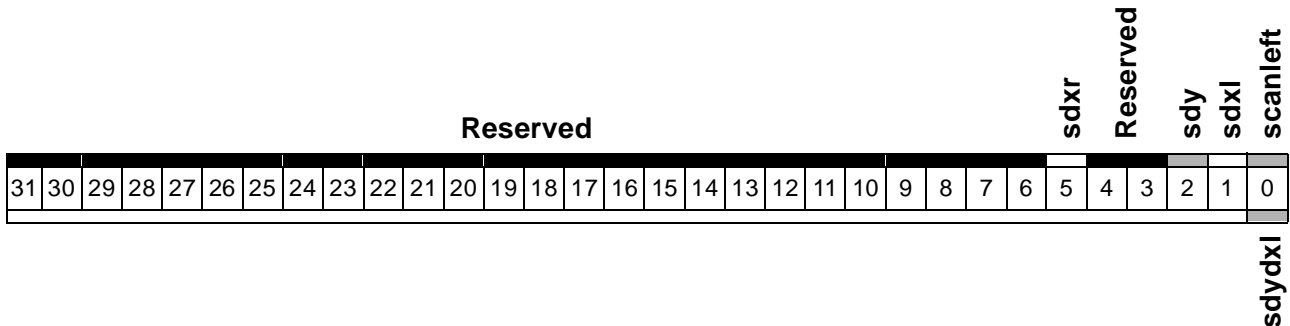
**WARNING! A soft reset will not re-read the chip strapping.**

**Reserved**  
**<31:1>**

Reserved. Writing to this field has no effect. Reading will give 0's.

**SGN****Sign**

<b>Address</b>	<b>mgabase1</b> + 1C58h (MEM)
<b>Attributes</b>	WO, FIFO, DYNAMIC, DWORD
<b>Reset Value</b>	Unknown



**Note:** Writing to this register when **DWGCTL**'s **sgnzero** bit = 1 will produce unpredictable results. Make sure that a '0' is written to **sgnzero** prior to accessing **SGN**.

- sdyl**  
<0> Sign of delta y minus delta x. This bit is shared with **scanleft**. It is defined for LINE drawing only and specifies the major axis. This bit is automatically initialized during AUTOLINE operations.
- 0: major axis is y
  - 1: major axis is x
- scanleft**  
<0> Horizontal scan direction left (1) vs. right (0). This bit is shared with **sdyl** and affects TRAPs and BLITs; **scanleft** is set according to the x scanning direction in a BLIT. Normally, this bit is always programmed to zero except for BITBLT when **bltmod** = BFCOL (see **DWGCTL** on page 4-40). For TRAP drawing, this bit must be set to 0 (scan right).
- sdxl**  
<1> Sign of delta x (line draw or left trapezoid edge). The **sdxl** field specifies the x direction for a line draw (**opcod** = LINE) or the x direction when plotting the left edge in a filled trapezoid draw. This bit is automatically initialized during AUTOLINE operations.
- 0: delta x is positive
  - 1: delta x is negative
- sdyl**  
<2> Sign of delta y. The **sdyl** field specifies the y direction of the destination address. This bit is automatically initialized during AUTOLINE operations. This bit should be programmed to zero for TRAP.
- 0: delta y is positive
  - 1: delta y is negative
- sdxr**  
<5> Sign of delta x (right trapezoid edge). The **sdxr** field specifies the x direction of the right edge of a filled trapezoid.
- 0: delta x is positive
  - 1: delta x is negative
- Reserved:** <4:3> <31:6>

Reserved. Writing to these fields has no effect.

## Funnel Shifter Control

SHIFT

<b>Address</b>	<b>mgabase1</b> + 1C50h (MEM)
<b>Attributes</b>	WO, FIFO, DYNAMIC, DWORD
<b>Reset Value</b>	Unknown

Reserved							stylelen						Reserved							funcnt											
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									funoff						Reserved							y_off			x_off						

**Note:** Writing to this register when the **DWGCTL** register's **shftzero** bit = 1 will produce unpredictable results. Make sure that a '0' has been written to **shftzero** prior to accessing **SHIFT**.

- funcnt**  
**<6:0>**
- Funnel count value. This field is used to drive the funnel shifter bit selection.
- For LINE operations, this is a countdown register. For 3D vectors, this field must be initialized to 0.
- This field will be modified during Blit operations.
- x\_off**  
**<3:0>**
- Pattern x offset. This field is used for TRAP operations without depth, to specify the x offset in the pattern. This offset must be in the range 0-7 (bit 3 is always 0).
- This field will be modified during Blit operations.
- y\_off**  
**<6:4>**
- Pattern y offset. This field is used for TRAP operations without depth, to specify the y offset in the pattern.
- This field will be modified during Blit operations.
- funoff**  
**<21:16>**
- Funnel shifter offset. For Blit operations, this field is used to specify a bit offset in the funnel shifter count. In this case **funoff** is interpreted as a 6-bit signed value.
- stylelen**  
**<22:16>**
- Line style length. For LINE operations, this field specifies the linestyle length. It indicates a location in the **SRC** registers (see [page 4-64](#)), so its value is the number of bits in the complete pattern minus one. For 3D vectors, this field must be initialized to 0.
- Reserved:** **<15:7>** **<31:23/22>**
- Reserved. Writing to these fields has no effect.

**SRC0, SRC1, SRC2, SRC3****Source**

<b>Address</b>	mgabase1 + 1C30h, + 1C34h, + 1C38h, + 1C3Ch (MEM)
<b>Attributes</b>	WO, FIFO, DYNAMIC, DWORD
<b>Reset Value</b>	Unknown

srcreg3								srcreg2								srcreg1								srcreg0								
127								96	95							64	63							32	31							0

**srcreg  
<127:0>**

Source register. The source register is used as source data for all drawing operations.

For LINE with the RPL or RSTR attribute, the source register is used to store the line style. The **funcnt** field of the **SHIFT** register points to the selected source register bit being used as the linestyle for the current pixel. Refer to Section 5.5.3.3 on page 5-19 for more details.

For TRAP with the RPL or RSTR attribute, the source register is used to store an 8 × 8 pattern (the odd bytes of the SRC registers must be a copy of the even bytes). Refer to Section 5.5.4.3 on page 5-24 for more details.

For all BITBLT operations, and for TRAP or LINE using depth mode, the source register is used internally for intermediate data.

A write to the **PAT** registers (see page 4-58) will load the **SRC** registers.

Status	STATUS
<b>Address</b>	<b>mgabase1</b> + 1E14 (MEM)
<b>Attributes</b>	RO, DYNAMIC, BYTE/WORD/DWORD
<b>Reset Value</b>	0000 0000 0000 0000 0000 0000 0?00 0000 b

Reserved																dwgengsts	Reserved																extpen	vlinepen	vsyncpen	vsyncsts	pickpen	Reserved
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							

- pickpen** <2> Pick interrupt pending. When set to ‘1’, indicates that a pick interrupt has occurred. This bit is cleared through the **pickiclr** bit (see **ICLEAR** on page 4-52) or upon soft or hard reset.
- vsyncsts** <3> VSYNC status. Set to ‘1’ during the VSYNC period. This bit follows the VSYNC signal.
- vsyncpen** <4> VSYNC interrupt pending. When set to ‘1’, indicates that a VSYNC interrupt has occurred. (This bit is a copy of the **crtcintCRT** field of the **INSTS0** VGA register). This bit is cleared through the **vintclr** bit of the **CRTC11** VGA register) or upon hard reset.
- vlinepen** <5> Vertical line interrupt pending. When set to ‘1’, indicates that the vertical line counter has reached the value of the vertical interrupt line count. See the **CRTC18** register on page 4-115. This bit is cleared through the **vlineiclr** bit (see **ICLEAR** on page 4-52) or upon soft or hard reset.
- extpen** <6> External interrupt pending. When set to ‘1’, indicates that the external interrupt line is driven. This bit is cleared by conforming to the interrupt clear protocol of the external device that drive the **EXTINT/** line. After a hard reset, the state of this bit is unknown (as indicated by the question mark in the ‘Reset Value’ above), as it depends on the state of the **EXTINT/** pin during the hard reset.
- dwgengsts** <16> Drawing engine status. Set to ‘1’ when the drawing engine is busy (a busy condition will be maintained until the BFIFO is empty, the drawing engine is finished with the last drawing command, and the memory controller has completed the last memory access).
- Reserved:** <1:0> <15:7> <31:17>  
Reserved. Writing to these fields has no effect. Reading will give 0’s.

A sample and hold circuit has been added to provide a correct, non-changing value during the full PCI read cycle (the status values are sampled at the start of the PCI access).

**VCOUNT****Vertical Count**

**Address**            **mgabase1 + 1E20 (MEM)**  
**Attributes**        **RO, DYNAMIC, WORD/DWORD**  
**Reset Value**        **Unknown**

**Reserved****vcount**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

**vcount**            Vertical counter value. Writing has no effect. Reading will give the current vertical count value.  
**<11:0>**

**Reserved**            Reserved. Writing to this field has no effect. Reading will give 0's.  
**<31:12>**

This register must be read using a word or dword access, because the value might change between two byte accesses. A sample and hold circuit will ensure a stable value for the duration of one PCI read access.

**X Destination Address****XDST**

<b>Address</b>	<b>mgabase1</b> + 1CB0h (MEM)
<b>Attributes</b>	WO, FIFO, DYNAMIC, DWORD
<b>Reset Value</b>	Unknown

**Reserved****xdst**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

**xdst**  
**<15:0>**

The x coordinate of destination address. The **xdst** field contains the running x coordinate (in pixels) of the destination address. It is a 16-bit signed value in two's complement notation.

- Before starting a vector draw, **xdst** must be loaded with the x coordinate of the starting point of the vector. At the end of a vector, **xdst** contains the address of the last pixel of the vector. This can also be done by accessing the **XYSTRT** register.
- This register does not require initialization for polyline operations.
- For trapezoids and BLITs, this register is automatically loaded from **fxleft** (see **FXLEFT** on page 4-50) and **fxright** (see **FXRIGHT** on page 4-51), and no initial value must be loaded.

**Reserved**  
**<31:16>**

Reserved. Writing to this field has no effect.

**XYEND****XY End Address**

<b>Address</b>	<b>mgabase1</b> + 1C44h (MEM)
<b>Attributes</b>	WO, FIFO, DYNAMIC, DWORD
<b>Reset Value</b>	Unknown

**y\_end****x\_end**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

The **XYEND** register is not a physical register. It is simply an alternate way to load registers **AR0** and **AR2**.

The **XYEND** register is only used for AUTOLINE drawing.

When **XYEND** is written, the following registers are affected:

- **x\_end**<15:0> --> **ar0**<17:0> (sign extended)
- **y\_end**<15:0> --> **ar2**<17:0> (sign extended)

**x\_end**  
<15:0>

The **x\_end** field contains the x coordinate of the end point of the vector. It is a 16-bit signed value in two's complement notation.

**y\_end**  
<31:16>

The **y\_end** field contains the y coordinate of the end point of the vector. It is a 16-bit signed value in two's complement notation.



**XY Start Address****XYSTRT**

<b>Address</b>	<b>mgabase1</b> + 1C40h (MEM)
<b>Attributes</b>	WO, FIFO, DYNAMIC, DWORD
<b>Reset Value</b>	Unknown

<b>y_start</b>																<b>x_start</b>															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

The **XYSTRT** register is not a physical register. It is simply an alternate way to load registers **AR5**, **AR6**, **XDST**, and **YDST**.

The **XYSTRT** register is only used for LINE and AUTOLINE. **XYSTRT** does not need to be initialized for polylines because all the registers affected by **XYSTRT** are updated to the endpoint of the vector at the end of the AUTOLINE.

When **XYSTRT** is written, the following registers are affected:

- **x\_start**<15:0> --> **xdst**<15:0>
- **x\_start**<15:0> --> **ar5**<17:0> (sign extended)
- **y\_start**<15:0> --> **ydst**<21:0> (sign extended) 0 --> **sellin**
- **y\_start**<15:0> --> **ar6**<17:0> (sign extended)

**x\_start**  
<15:0> The **x\_start** field contains the x coordinate of the starting point of the vector. It is a 16-bit signed value in two's complement notation.

**y\_start**  
<31:16> The **y\_start** field contains the y coordinate of the starting point of the vector. This coordinate is always xy (this means that in order to use the **XYSTRT** register the linearizer must be used). It is a 16-bit signed value in two's complement notation.

**YBOT****Clipper Y Maximum Boundary**

<b>Address</b>	<b>mgabase1</b> + 1C9Ch (MEM)
<b>Attributes</b>	WO, FIFO, STATIC, DWORD
<b>Reset Value</b>	Unknown

Reserved

**cybot**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

**cybot**  
**<22:0>**

Clipper y bottom boundary. The **cybot** field contains an unsigned 23-bit value which is interpreted as a positive pixel address and compared with the current **ydst** (see **YDST** on page 4-71). The value of the **ydst** field must be less than or equal to **cybot** to be inside the drawing window.

This register must be programmed with a linearized line number:

$$\mathbf{cybot} = (\text{bottom line number}) \times \mathbf{PITCH} + \mathbf{YDSTORG}$$

The **YBOT** register must be loaded with a multiple of 32 (the five LSBs = 0). There is no way to disable clipping.

**Reserved**  
**<31:23>**

Reserved. Writing to this field has no effect.

## Y Address

## YDST

<b>Address</b>	<b>mgabase1</b> + 1C90h (MEM)
<b>Attributes</b>	WO, FIFO, DYNAMIC, DWORD
<b>Reset Value</b>	Unknown

sellin			Reserved																			ydst									
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**ydst**  
**<21:0>**

The y destination. The **ydst** field contains the current y coordinate (in pixels) of the destination address as a signed value in two's complement notation. Two formats are supported: linear format and xy format. The current format is selected by **ylin** (see **PITCH** on page 4-59).

When xy format is used (**ylin**=0), ydst represents the y coordinate of the address. The valid range is -32768 to +32767 (16-bit signed). The xy value is always converted to a linear value before being used.

When linear format is used (**ylin**=1), ydst must be programmed as follows:

$$\mathbf{ydst} \leftarrow (\text{Y coordinate}) * \mathbf{PITCH} \gg 5$$

The y coordinate range is from -32768 to +32767 (16-bit signed) and the pitch range is from 32 to 2048. Pitch is also a multiple of 32.

- Before starting a vector draw, **ydst** must be loaded with the y coordinate of the starting point of the vector. This can be done by accessing the **XYSTRT** register. This register does not require initialization for polyline operations.
- Before starting a BLIT, **ydst** is loaded with the y coordinate of the starting corner of the destination rectangle.
- For trapezoids, this register must be loaded with the y coordinate of the first scanned line of the trapezoid.

**Reserved**  
**<28:22>**

Reserved. Writing to this field has no effect.

**sellin**  
**<31:29>**

Selected line. The **sellin** field is used to perform the dithering, patterning, and transparency functions. During linearization, this field is loaded with the three LSBs of **ydst**. If no linearization occurs, then those bits must be initialized correctly if one of the above-mentioned functions is to be used.

**YDSTLEN****Y Destination and Length**

<b>Address</b>	<b>mgabase1</b> + 1C88h (MEM)
<b>Attributes</b>	WO, FIFO, STATIC, DWORD
<b>Reset Value</b>	Unknown

<b>yval</b>																<b>length</b>															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

The **YDSTLEN** register is not a physical register. It is simply an alternate way to load the **YDST** and **LEN** registers.

**length**  
**<15:0>** Length. See the **LEN** register on page 4-54.

**yval**  
**<31:16>** The y destination value. See the **YDST** register on page 4-71. The **yval** field can be used to load the **YDST** register in xy format. In this case the valid range -32768 to +32767 (16-bit signed) for **YDST** is respected.

**ydst<21:0>** <= sign extension (**yval<31:16>**)

For the linear format, **yval** does not contain enough bits, so **YDST** must be used directly.

## Memory Origin

## YDSTORG

<b>Address</b>	<b>mgabase1</b> + 1C94h (MEM)
<b>Attributes</b>	WO, FIFO, STATIC, DWORD
<b>Reset Value</b>	Unknown

Reserved									ydstorg																						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**ydstorg**  
**<22:0>** Destination y origin. The **ydstorg** field is a 23-bit unsigned value. It gives an offset value in pixel units, used to position the first pixel of the first line of the screen. This register is used to initialize the **YDST** address.

This register must be loaded with a multiple of values according to the table below due to a restriction involving block mode (see ‘Constant Shaded Trapezoids / Rectangle Fills’ on page 5-23) and fast blits (see ‘Two-operand Fast Bitblts’ on page 5-30). Refer also to page 4-41 for additional restrictions that apply to block mode (**atype** = BLK).

<b>pwidth</b>	<i>Non-Intereave</i>	<i>Interleave</i>
PW8	64	128
PW16	32	64
PW24	64	128
PW32	32	32

**Reserved**  
**<31:23>** Reserved. Writing to this field has no effect.

## YTOP

## Clipper Y Top Boundary

<b>Address</b>	<b>mgabase1</b> + 1C98h (MEM)
<b>Attributes</b>	WO, FIFO, STATIC, DWORD
<b>Reset Value</b>	Unknown

Reserved					cytop																										
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**cytop**  
**<22:0>** Clipper y top boundary. The **cytop** field contains an unsigned 23-bit value which is interpreted as a positive pixel address and compared with the current **ydst** (see **YDST** on page 4-71). The value of the **ydst** field must be greater than or equal to **cytop** to be inside the drawing window.

This register must be programmed with a linearized line number:

$$\mathbf{cytop} = (\text{top line number}) \times \mathbf{PITCH} + \mathbf{YDSTORG}$$

This register must be loaded with a multiple of 32 (the five LSBs = 0).

Note that since the **cytop** value is interpreted as positive, any negative **ydst** value is automatically outside the clipping window.

There is no way to disable clipping.

**Reserved**  
**<31:23>** Reserved. Writing to this field has no effect.

**Z-Depth Origin****ZORG**

**Address**            **mgabase1** + 1C0Ch (MEM)  
**Attributes**        WO, FIFO, STATIC, DWORD  
**Reset Value**        Unknown

Reserved									zorg																						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**zorg**            Z-depth origin. The **zorg** field is a 23-bit unsigned value which provides an offset value  
**<22:0>**        (the Base Address) in order to position the first pixel in the z-depth buffer.

The **zorg** field corresponds to a byte address in memory. This register must be set so that there is no overlap with the frame buffer.

This field must be loaded with a multiple of 512 (the nine LSBs = 0).

$$\mathbf{zorg} = \mathbf{Z\ depth\ origin} - \mathbf{ydstorg} * 2$$

**Reserved**        Reserved. Writing to this field has no effect.  
**<31:23>**

*This page is intentionally blank.*



## 4.2 VGA Mode Registers

The MGA-2064W VGA mode register descriptions contain a (single-underlined) main header which indicates the register's name and mnemonic. Below the main header, the memory address or index, attributes, and reset value are indicated. Next, an illustration of the register identifies the bit fields, which are then described in detail below the illustration. The reserved bit fields are underscored by black bars, and all other fields are delimited by alternating white and gray bars.

### Sample VGA Mode Register Description

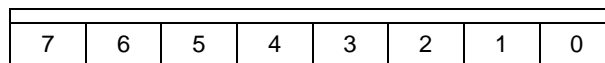
**SAMPLE\_VGA**

**Address** <value> (I/O), <value> (MEM)  
**Attributes** R/W, BYTE/WORD, STATIC  
**Reset Value** <value>

↖ Main header

↖ Underscore bar

field



### Address

This address is an offset from the Power Graphic mode base memory address. The memory addresses can be read, write, color, or monochrome, as indicated.

### Index

The index is an offset from the starting address of the register group.

### Attributes

The VGA mode attributes are:

- RO            There are no writable bits.
- WO:         The state of the written bits cannot be read.
- R/W:        The state of the written bits can be read.
- BYTE:      8-bit access to the register is possible.
- WORD:      16-bit access to the register is possible.
- STATIC:     The contents of the register will not change during an operation.
- DYNAMIC:   The contents of the register might change during an operation.

### Reset Value

The reset values for the VGA mode registers are expressed as binary values. The letter 'n' in a reset value indicates an indexed register bit (refer to the indexed register for the reset value).

- 000X 0000 b (b = Binary, X = undefined)
- 0X00 0000 b (00 = reset on soft and hard reset; see above)

**ATTR****Attribute Controller**

**Address** R/W at port 03C0h (I/O), **mgabase1** + 1FC0h (MEM) VGA  
 R at port 03C1h (I/O), **mgabase1** + 1FC1h (MEM) VGA

**Attributes** BYTE, STATIC

**Reset Value** nnnn nnnn 0000 0000 b

attrd								Reserved pas			attrx				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**attrx** Attribute controller index register. VGA.

**<4:0>**

A binary value that points to the VGA Attribute Controller register where data is to be written or read.

<i>Register name</i>	<i>Mnemonic</i>	<i>attrx address</i>
Palette entry 0	<b>ATTR0</b>	00h
Palette entry 1	<b>ATTR1</b>	01h
Palette entry 2	<b>ATTR2</b>	02h
Palette entry 3	<b>ATTR3</b>	03h
Palette entry 4	<b>ATTR4</b>	04h
Palette entry 5	<b>ATTR5</b>	05h
Palette entry 6	<b>ATTR6</b>	06h
Palette entry 7	<b>ATTR7</b>	07h
Palette entry 8	<b>ATTR8</b>	08h
Palette entry 9	<b>ATTR9</b>	09h
Palette entry A	<b>ATTRA</b>	0Ah
Palette entry B	<b>ATTRB</b>	0Bh
Palette entry C	<b>ATTRC</b>	0Ch
Palette entry D	<b>ATTRD</b>	0Dh
Palette entry E	<b>ATTRE</b>	0Eh
Palette entry F	<b>ATTRF</b>	0Fh
Attribute Mode Control	<b>ATTR10</b>	10h
Overscan Color	<b>ATTR11</b>	11h
Color Plane Enable	<b>ATTR12</b>	12h
Horizontal Pel Panning	<b>ATTR13</b>	13h
Color Select	<b>ATTR14</b>	14h
Reserved (1)		15-1F

(1) Writing to a reserved index has no effect.

- A read from port 3BA/3DAh resets this port to the attributes address register. The first write at 3C0 after a 3BA/3DAh reset accesses the attribute index. The next write at 3C0 accesses the palette. Subsequent writes at 3C0 toggle between the index and the palette.
- A read at port 3C1 does not toggle the index/data pointer.

## Attribute Controller

... ATTR

**Example of a palette write:**

Reset pointer:	read at port 3BA
Write index:	write at port 3C0
Write color:	write at port 3C0

**Example of a palette read:**

Reset pointer:	read at port
3BA Write index:	write at port
3C0 Read color:	read at port 3C1

**pas**  
<5>

Palette address source. VGA.

This bit controls use of the internal palette. If **pas** = 0, the host CPU can read and write the palette, and the display is forced to the overscan color. If **pas** = 1, the palette is used normally by the video stream to translate color indices (CPU writes are inhibited and reads return all '1's). Normally, the internal palette is loaded during the blank time, since loading inhibits video translation.

**Reserved**  
<7:6>

Writing has no effect. Reading will give 0's.

**attrd**  
<15:8>**ATTR** data register.Retrieve or write the contents of the register pointed to by the **attrx** field.

**ATTR0 to ATTRF****Palette Entry 0h to Fh**

**Index**            **attrx = 00h to attrx = 0Fh**  
**Reset Value**    0000 0000 b

Reserved			palet0-F				
7	6	5	4	3	2	1	0

**palet0-F**  
**<5:0>**

Internal palette data. VGA.

These six-bit registers allow dynamic mapping between the text attribute or graphic color input value and the display color on the CRT screen. These internal palette register values are sent from the chip to the video DAC, where they in turn serve as addresses to the DAC internal registers. A palette register can be loaded only when **pas** (**ATTR**<5>) = 0.

**Reserved**  
**<7:6>**

Writing has no effect. This field returns all zeroes when read.

## Attribute Mode Control

ATTR10

Index attrx = 10h  
Reset Value 0000 0000 b

	p5p4	pelwidth	pancomp	Reserved	blinken	lgren	mono	atcgrmode
	7	6	5	4	3	2	1	0

**atcgrmode**  
<0> Graphic/alphanumeric mode. VGA.

- 0: Alphanumeric mode is enabled and the input of the internal palette circuit comes from the expansion of the foreground/background attribute.
- 1: Graphics mode is enabled and the input of the internal palette comes from the frame buffer pixel. This bit also selects between graphics blinking or character blinking if blinking is enabled (**blinken** = 1).

**mono**<1> Mono emulation. VGA.

- 0: Color emulation.
- 1: Monochrome emulation.

**lgren**<2> Enable line graphics character code. VGA.

- 0: The ninth dot of a line graphic character (a character between C0h and DFh) will be the same as the background.
- 1: Forces the ninth dot to be identical to the eighth dot of the character. For other ASCII codes, the ninth dot will be the same as the background.

For character fonts that do not utilize the line graphics character, lgren should be '0'. Otherwise, unwanted video information will be displayed. This bit is 'don't care' in graphics modes (**atcgrmode** = 0).

**blinken**  
<3> Select background intensity or blink enable. VGA.

- 0: Blinking is disabled. In alpha modes (**atcgrmode** (**ATTR10**<0>) = 0), this bit defines the attribute bit 7 as a background high-intensity bit. In graphic modes, planes 3 to 0 select 16 colors out of 64.
- 1: Blinking is enabled. In alpha modes (**atcgrmode** = 0), this bit defines the attribute bit 7 as a blink attribute (when the attribute bit 7 is '1', the character will blink). The blink rate of the character is vsync/32, and the blink duty cycle is 50%. In monochrome graphics mode (**mono** and **atcgrmode** (**ATTR10**<1:0>) = 11), all pixels toggle on and off. In color graphics modes (**mono** and **atcgrmode** (**ATTR10**<1:0>) = 01), only pixels that have **blinken** (bit 3) high will toggle on and off: other pixels will have their bit 3 forced to '1'. The graphic blink rate is VSYNC/32. Graphic blink logic is applied after plane masking (that is, if plane 3 is disabled, monochrome mode will blink and color mode will not blink).

**Reserved**  
<4> Writing has no effect. This field returns zero when read.

## ... ATTR10

## Attribute Mode Control

- pancomp**  
<5> Pel panning compatibility. VGA.
- 0: Line compare has no effect on the output of the PEL panning register.
  - 1: A successful line compare in the CRT controller maintains the panning value to 0 until the end of frame (until next vsync), at which time the panning value returns to the value of **hpelcnt** (**ATTR13**<3:0>). This bit allows panning of only the top portion of the display.
- pelwidth**  
<6> Pel width. VGA.
- 0: The six bits of the internal palette are used instead.
  - 1: Two 4-bit sets of video data are assembled to generate 8-bit video data.
- p5p4**  
<7> P5/P4 select. VGA.
- 0: Bits 5 and 4 of the internal palette registers are transmitted to the RAMDAC.
  - 1: When it is set to '1', **colsel54** (**ATTR14**<1:0>) will be transmitted to the RAM-DAC. See the **ATTR14** register on page 4-86.

**Overscan Color****ATTR11**

**Index** attrx = 11h  
**Reset Value** 0000 0000 b

**ovscol**

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

**ovscol**  
**<7:0>**

Overscan color. VGA.

Determines the overscan (border) color displayed on the CRT screen. The value programmed is the index of the border color in the RAMDAC. The border color is displayed when the internal DISPEN signal is inactive and blank is not active.

## ATTR12

## Color Plane Enable

**Index** attrx = 12h  
**Reset Value** 0000 0000 b

Res.		vidstmx		colplen			
7	6	5	4	3	2	1	0

**colplen**  
**<3:0>** Enable color plane. VGA.

**vidstmx**  
**<5:4>** Video status multiplexer (MUX). VGA.

These bits select two of eight color outputs for the status port. Refer to the table in the description of the **INSTS1** register's **diag** field that appears on [page 4-141](#).

**Reserved**  
**<7:6>** Writing has no effect. This field returns all zeroes when read.



## Horizontal Pel Panning

ATTR13

Index attrx = 13h  
 Reset Value 0000 0000 b

Reserved				hpelcnt			
7	6	5	4	3	2	1	0

**hpelcnt**  
**<3:0>**

Horizontal pel count. VGA.

This 4-bit value specifies the number of picture elements to shift the video data horizontally to the left, according to the following table (values 9 to 15 are reserved):

<b>hpelcnt</b>	8 dot mode pixel shifted <b>dotmode</b> (SEQ1<0>) = 1	9 dot mode pixel shifted <b>dotmode</b> = 0	<b>mode256</b> (GCTL5<6>) = 1
0	0	1	0
1	1	2	-
2	2	3	1
3	3	4	-
4	4	5	2
5	5	6	-
6	6	7	3
7	7	8	-
8	-	0	-

**Reserved**  
**<7:4>**

Writing has no effect. This field returns all zeroes when read.

## ATTR14

## Color Select

**Index** attrx = 14h  
**Reset Value** 0000 0000 b

Reserved				colsel76		colsel54	
7	6	5	4	3	2	1	0

**colsel54**  
 <1:0> Select color 5 to 4. VGA.

When **p5p4** (**ATTR10**<7>) is '1', **colsel54** is used instead of internal palette bits 5 and 4. This mode is intended for rapid switching between sets of colors (four sets of 16 colors can be defined). These bits are 'don't care' when **mode256** = 1.

**colsel76**  
 <3:2> Select color 7 to 6. VGA.

These bits are the two MSB bits of the the external color palette index. They can rapidly switch between four sets of 64 colors. These bits are 'don't care' when **mode256** (**GCTL5**<6>) = 1.

**Reserved**  
 <7:4> Writing has no effect. This field returns all zeroes when read.

**CRTC Registers****CRTC**

<b>Address</b>	03B4h (I/O), <b>mgabase1</b> + 1FB4h (MEM) ( <b>MISC</b> <0> == 0: MDA emulation) 03D4h (I/O), <b>mgabase1</b> + 1FD4h (MEM) ( <b>MISC</b> <0> == 1: CGA emulation)
<b>Attributes</b>	R/W, BYTE/WORD, STATIC
<b>Reset Value</b>	nnnn nnnn 0000 0000 b

<b>crtcd</b>								<b>Reserved</b>				<b>crtcX</b>			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**crtcX** CRTC index register.

**<5:0>**

A binary value that points to the VGA **CRTC** register where data is to be written or read when the **crtcd** field is accessed.

<i>Register name</i>	<i>Mnemonic</i>	<i>crtcX address</i>
CRTC register index	<b>CRTCx</b>	--
Horizontal Total	<b>CRTC0</b>	00h
Horizontal Display Enable End	<b>CRTC1</b>	01h
Start Horizontal Blanking	<b>CRTC2</b>	02h
End Horizontal Blanking	<b>CRTC3</b>	03h
Start Horizontal Retrace Pulse	<b>CRTC4</b>	04h
End Horizontal Retrace	<b>CRTC5</b>	05h
Vertical Total	<b>CRTC6</b>	06h
Overflow	<b>CRTC7</b>	07h
Preset Row Scan	<b>CRTC8</b>	08h
Maximum Scan Line	<b>CRTC9</b>	09h
Cursor Start	<b>CRTCA</b>	0Ah
Cursor End	<b>CRTCB</b>	0Bh
Start Address High	<b>CRTCC</b>	0Ch
Start Address Low	<b>CRTCD</b>	0Dh
Cursor Location High	<b>CRTCE</b>	0Eh
Cursor Location Low	<b>CRTCF</b>	0Fh
Vertical Retrace Start	<b>CRTC10</b>	10h
Vertical Retrace End	<b>CRTC11</b>	11h
Vertical Display Enable End	<b>CRTC12</b>	12h
Offset	<b>CRTC13</b>	13h
Underline Location	<b>CRTC14</b>	14h
Start Vertical Blank	<b>CRTC15</b>	15h
End Vertical Blank	<b>CRTC16</b>	16h
CRTC Mode Control	<b>CRTC17</b>	17h
Line Compare	<b>CRTC18</b>	18h
Reserved - read as 0 (1)	----	19h - 21h
CPU Read Latch	<b>CRTC22</b>	22h
Reserved - read as 0	----	23h

(1) Writing to a reserved index has no effect.

## ... CRTC

## CRTC Registers

<i>Register name</i>	<i>Mnemonic</i>	<i>crtcx address</i>
Attribute address/data select	<b>CRTC24</b>	24h
Reserved - read as 0	----	25h
Attribute address	<b>CRTC26</b>	26h
Reserved -- read as 0	----	27h
Reserved -- read as 0	----	28h - 3Fh

**Reserved**  
<7:6>

Writing has no effect. Reading will give 0's.

**crtcd**  
<15:8>

CRTC data register.

Retrieve or write the contents of the register pointed to by the **crtcx** field.

**Horizontal Total****CRTC0**

**Index**            **crtcx** = 00h  
**Reset Value**    0000 0000 b

**htotal**

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

**htotal  
<7:0>**

Horizontal total. VGA/MGA.

This is the low-order eight bits of a 9-bit register (bit 8 is contained in **htotal** (**CRTCEXT1**<0>)). This field defines the total horizontal scan period in character clocks, minus 5.

This register can be write-inhibited when **crtcprotect** (**CRTC11**<7>) = 1.

**CRTC1****Horizontal Display Enable End**

**Index**            **crtcx** = 01h  
**Reset Value**    0000 0000 b

**hdispend**

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

**hdispend  
<7:0>**

Horizontal display enable end. VGA/MGA.

Determines the number of displayed characters per line. The display enable signal becomes inactive when the horizontal character counter reaches this value.

This register can be write-inhibited when **crtcprotect** (**CRTC11**<7>) = 1.

**Start Horizontal Blanking****CRTC2**

**Index**            **crtcx = 02h**  
**Reset Value**    **0000 0000 b**

**hblkstr**

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

**hblkstr**  
**<7:0>**

Start horizontal blanking. VGA/MGA.

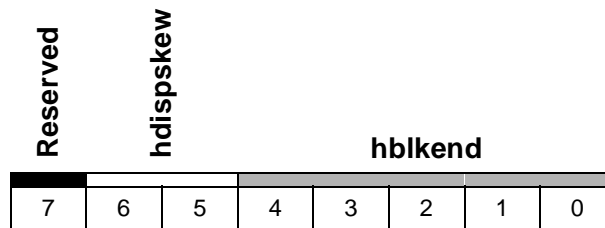
This is the low-order eight bits of a 9-bit register. Bit 8 is contained in **hblkstr** (**CRTCEXT1**<1>). The horizontal blanking signal becomes active when the horizontal character counter reaches this value.

This register can be write-inhibited when **crtcprotect** (**CRTC11**<7>) = 1.

## CRTC3

## End Horizontal Blanking

Index **crtcx** = 03h  
 Reset Value 1000 0000 b



**hblkend**  
**<4:0>**

End horizontal blanking bits. VGA/MGA.

The horizontal blanking signal becomes inactive when, after being activated, the lower six bits of the horizontal character counter reach the horizontal blanking end value. The five lower bits of this value are located here; bit 5 is located in the **CRTC5** register, and bit 6 is located in **CRTCEXT1**.

This register can be write-inhibited when **crtcprotect** (**CRTC11**<7>) = 1.

**hdispskew**  
**<6:5>**

Display enable skew control. VGA/MGA.

Defines the number of character clocks to delay the display enable signal to compensate for internal pipeline delays.

Normally, the hardware can accommodate the delay, but the VGA design allows greater flexibility by providing extra control.

<b>hdispskew</b>	<i>Skew</i>
00	0 additional character delays
01	1 additional character delays
10	2 additional character delays
11	3 additional character delays

**Reserved**  
**<7>**

This field is defined as a bit for chip testing on the IBM VGA, but is not used on the MGA. Writing to it has no effect (it will read as 1). For compatibility considerations, a 1 should be written to it.



**Start Horizontal Retrace Pulse****CRTC4**

**Index**            **crtc<sub>x</sub>** = 04h  
**Reset Value**    0000 0000 b

**hsyncstr**

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

**hsyncstr**  
**<7:0>**

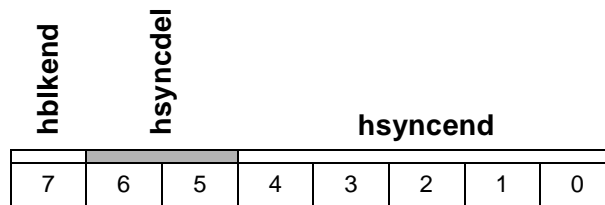
Start horizontal retrace pulse. VGA/MGA.

These are the low-order eight bits of a 9-bit register. Bit 8 is contained in **hsyncstr** (**CRTCEXT1**<2>). The horizontal sync signal becomes active when the horizontal character counter reaches this value.

This register can be write-inhibited when **crtcprotect** (**CRTC11**<7>) = 1.

**CRTC5****End Horizontal Retrace**

**Index**            **crtcx = 05h**  
**Reset Value**    **0000 0000 b**



**hsyncend**  
**<4:0>**

End horizontal retrace. VGA/MGA.

The horizontal sync signal becomes inactive when, after being activated, the five lower bits of the horizontal character counter reach the end horizontal retrace value.

This register can be write-inhibited when **crtcprotect** (**CRTC11**<7>) = 1.

**hsyncdel**  
**<6:5>**

Horizontal retrace delay. VGA/MGA.

Defines the number of character clocks that the hsync signal is delayed to compensate for internal pipeline delays.

<b>hsyncdel</b>	<i>Skew</i>
00	0 additional character delays
01	1 additional character delays
10	2 additional character delays
11	3 additional character delays

**hblkend**  
**<7>**

End horizontal blanking bit 5. VGA/MGA.

Bit 5 of the End Horizontal Blanking value. See the **CRTC3** register on page 4-92.

## Vertical Total

CRTC6

**Index**            **crtc<sub>x</sub>** = 06h  
**Reset Value**    0000 0000 b

**vtotal**

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

**vtotal  
<7:0>**

Vertical total. VGA/MGA.

These are the low-order eight bits of a 12-bit register. Bit 8 is contained in **CRTC7**<0>, bit 9 is in **CRTC7**<5>, and bits 10 and 11 are in **CRTCEXT2**<1:0>. The value defines the vsync period in scan lines if **hsyncsel** (**CRTC17**<2>) = 0, or in double scan lines if **hsyncsel** = 1).

This register can be write-inhibited when **crtcprotect** (**CRTC11**<7> = 1).

## CRTC7

## Overflow

Index **crtcx = 07h**  
 Reset Value 0000 0000 b

<b>vsyncstr</b>	<b>vdispend</b>	<b>vtotal</b>	<b>linecomp</b>	<b>vblkstr</b>	<b>vsyncstr</b>	<b>vdispend</b>	<b>vtotal</b>
7	6	5	4	3	2	1	0

- vtotal**  
 <0> Vertical total bit 8. VGA/MGA.  
 Contains bit 8 of the Vertical Total. See the **CRTC6** register on page 4-95.  
 This register can be write-inhibited when **crtcprotect** (**CRTC11**<7>) = 1, except for **linecomp**.
- vdispend**  
 <1> Vertical display enable end bit 8. VGA/MGA.  
 Contains bit 8 of the Vertical Display Enable End. See the **CRTC12** register on page 4-107.
- vsyncstr**  
 <2> Vertical retrace start bit 8. VGA/MGA.  
 Contains bit 8 of the Vertical Retrace Start. See the **CRTC10** register on page 4-105.
- vblkstr**  
 <3> Start vertical blank bit 8. VGA/MGA.  
 Contains bit 8 of the Start Vertical Blank. See the **CRTC15** register on page 4-110.
- linecomp**  
 <4> Line compare bit 8. VGA/MGA.  
 Line compare bit 8. See the **CRTC18** register on page 4-115. This bit is not write-protected by **crtcprotect** (**CRTC11**<7>).
- vtotal**  
 <5> Vertical total bit 9. VGA/MGA.  
 Contains bit 9 of the Vertical Total. See the **CRTC6** register on page 4-95.
- vdispend**  
 <6> Vertical display enable end bit 9. VGA/MGA.  
 Contains bit 9 of the Vertical Display Enable End. See the **CRTC12** register on page 4-107.
- vsyncstr**  
 <7> Vertical retrace start bit 9. VGA/MGA.  
 Contains bit 9 of the Vertical Retrace Start. See the **CRTC10** register on page 4-105.

## Preset Row Scan

CRTC8

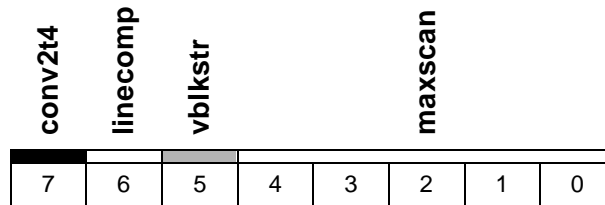
**Index**            **crtc<sub>x</sub>** = 08h  
**Reset Value**    0000 0000 b

Res.			bytepan					prowscan				
7	6	5	4	3	2	1	0					

- prowscan**  
**<4:0>**            Preset row scan. VGA/MGA.  
 After a vertical retrace, the row scan counter is preset with the value of **prowscan**. At maximum row scan compare time, the row scan is cleared (not preset). The units can be one or two scan lines:
- **conv2t4** (**CRTC9**<7>) = 0: 1 scan line
  - **conv2t4** = 1: 2 scan lines
- bytepan**  
**<6:5>**            Byte panning control. VGA/MGA.  
 This field controls the number of bytes to pan during a panning operation.
- Reserved**  
**<7>**                Writing has no effect. Reading will give 0's.

**CRTC9****Maximum Scan Line**

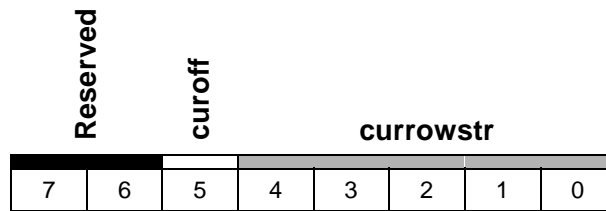
**Index**            **crtcx = 09h**  
**Reset Value**    **0000 0000 b**



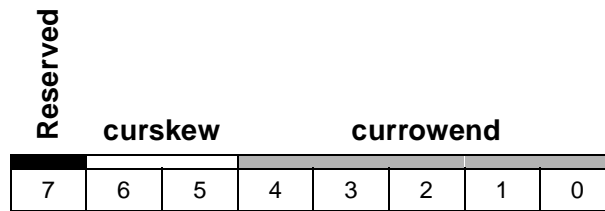
- maxscan**  
**<4:0>**            Maximum scan line. VGA/MGA.  
This field specifies the number of scan lines minus one per character row.
- vblkstr**  
**<5>**                Start vertical blank bit 9. VGA/MGA.  
Bit 9 of the Start Vertical Blank register. [See the CRTC15 register on page 4-110.](#)
- linecomp**  
**<6>**                Line compare bit 9. VGA/MGA.  
Bit 9 of the Line Compare register. [See the CRTC18 register on page 4-115.](#)
- conv2t4****<7>**        200 to 400 line conversion. VGA/MGA.  
Controls the row scan counter clock and the time when the start address latch loads a new memory address:
- **conv2t4 (CRTC9<7>) = 0:** HS
  - **conv2t4 = 1:** HS/2
- This feature allows a low resolution mode (200 lines, for example) to display as 400 lines on a display monitor. This lowers the requirements for sync capability of the monitor.

**Cursor Start****CRTCA**

**Index**            **crtcx = 0Ah**  
**Reset Value**    **0000 0000 b**



- currowstr <4:0>**      Row scan cursor begins. VGA.  
 These bits specify the row scan of a character line where the cursor is to begin.  
 When the cursor start register is programmed with a value greater than the cursor end register, no cursor is generated.
- curoff <5>**      Cursor off. VGA.
- Logical '1': turn off the cursor
  - Logical '0': turn on the cursor
- Reserved <7:6>**      Writing has no effect. This field returns all zeroes when read.

**CRTCx****Cursor End****Index**      **crtc<sub>x</sub>** = 0Bh**Reset Value**      0000 0000 b**currowend**      Row scan cursor ends. VGA.**<4:0>**

This field specifies the row scan of a character line where the cursor is to end.

**curskew**

Cursor skew control. VGA.

**<6:5>**

These bits control the skew of the cursor signal according to the following table:

<b>curskew</b>	<i>Skew</i>
00	0 additional character delays
01	Move the cursor right by 1 character clock
10	Move the cursor right by 2 character clocks
11	Move the cursor right by 3 character clocks

**Reserved****<7>**

Writing has no effect. This field returns zero when read.



**Start Address High****CRTCC**

**Index**            **crtcx = 0Ch**  
**Reset Value**    **0000 0000 b**

**startadd**

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

**startadd  
<7:0>**

High order start address. VGA/MGA.

These are the middle eight bits of the start address. The 20-bit value from the **startadd** (**CRTCEXT0**<3:0>) high-order and low-order start address registers is the first address after the vertical retrace on each screen refresh.

**Interleave configuration** (see page 6-7)

The start address must meet the following criteria:

- Obtain a DDWORD (64-bit) linear address. In 24 bit/pixel modes, the address must conform to modulo 24 format. In all other display modes, the address must conform to modulo 8 format.

**Non-interleave configuration**

The start address must meet the following criteria:

- Obtain a DWORD (32-bit) linear address. In 24 bit/pixel modes, the address must conform to modulo 12 format. In all other display modes, the address must conform to modulo 4 format.

The modulo value is obtained from the zoom factor.

**CRTCD****Start Address Low**

**Index**            **crtcx** = 0Dh  
**Reset Value**    0000 0000 b

**startadd**

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

**startadd**  
**<7:0>**

Low order start address. VGA/MGA.

These are the low-order eight bits of the start address. See the **CRTCC** register on page 4-101.

**Cursor Location High****CRTCE**

**Index**            **crtcx** = 0Eh  
**Reset Value**    0000 0000 b

**curloc**

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

**curloc**  
**<7:0>**

High order cursor location. VGA.

These are the high-order eight bits of the cursor address. The 16-bit bit value from the high-order and low-order cursor location registers is the character address where the cursor will appear. The cursor is available only in alphanumeric mode.

**CRTCF****Cursor Location Low**

**Index**            **crtcx** = 0Fh  
**Reset Value**    0000 0000 b

**curloc**

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

**curloc**  
**<7:0>**

Low order cursor location. VGA.

These are the low-order eight bits of the cursor location. See the **CRTCE** register on page 4-103.

**Vertical Retrace Start****CRTC10**

**Index**            **crtcx = 10h**  
**Reset Value**    **0000 0000 b**

**vsyncstr**

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

**vsyncstr**  
**<7:0>**

Vertical retrace start bits 7 to 0. VGA/MGA.

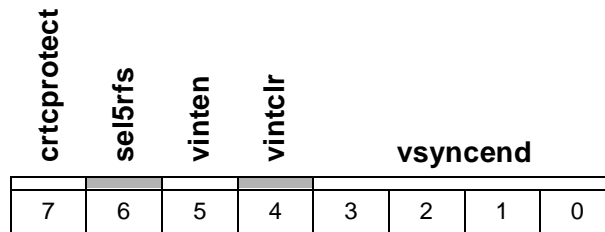
The vertical sync signal becomes active when the vertical line counter reaches the vertical retrace start value (a 12-bit value). The lower eight bits are located here. Bit 8 is in **CRTC7<2>**, bit 9 is in **CRTC7<7>**, and bits 10 and 11 are in **CRTCEXT2<6:5>**.

The units can be one or two scan lines:

- **hsyncsel** (**CRTC17<2>**) = 0: 1 scan line
- **hsyncsel** = 1: 2 scan lines

**CRTC11****Vertical Retrace End**

**Index**            **crtcx = 11h**  
**Reset Value**    **0000 0000 b**



- vsyncend**  
**<3:0>**            Vertical retrace end. VGA/MGA.  
 The vertical retrace signal becomes inactive when, after being activated, the lower four bits of the vertical line counter reach the vertical retrace end value.
- vintclr**  
**<4>**                Clear vertical interrupt. VGA/MGA.  
 A '0' in **vintclr** will clear the internal request flip-flop.  
 After clearing the request, an interrupt handler must write a '1' to **vintclr** in order to allow the next interrupt to occur.
- vinten**  
**<5>**                Enable vertical interrupt. VGA/MGA.
- 0: Enables a vertical retrace interrupt. If the interrupt request flip-flop has been set at enable time, an interrupt will be generated. We recommend setting **vintclr** to '0' when **vinten** is brought low.
  - 1: Removes the vertical retrace as an interrupt source.
- sel5rfs**  
**<6>**                Select 5 refresh cycles. VGA.  
 This bit is read/writable to maintain compatibility with the IBM VGA. It does not control the MGA RAM refresh cycle (as in the IBM implementation). Refresh cycles are optimized to minimize disruptions.
- crtcprotect**  
**<7>**                Protect **CRTC** registers 0-7. VGA/MGA.
- 1: Disables writing to **CRTC** registers 0 to 7.
  - 0: Enables writing. The **linecomp** (line compare) field of **CRTC7** is not protected.

**Vertical Display Enable End****CRTC12**

**Index**            **crtcx = 12h**  
**Reset Value**    **0000 0000 b**

**vdispend**

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

**vdispend  
<7:0>**

Vertical display enable end. VGA/MGA.

The vertical display enable end value determines the number of displayed lines per frame. The display enable signal becomes inactive when the vertical line counter reaches this value. Bits 7 to 0 are located here. Bit 8 is in **CRTC7<1>**, bit 9 is in **CRTC7<6>**, and bit 10 is in **CRTCEXT2<2>**.

**CRTC13****Offset**

**Index**            **crtcx = 13h**  
**Reset Value**    **0000 0000 b**

**offset**

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

**offset  
<7:0>**

Logical line width of the screen. VGA/MGA.

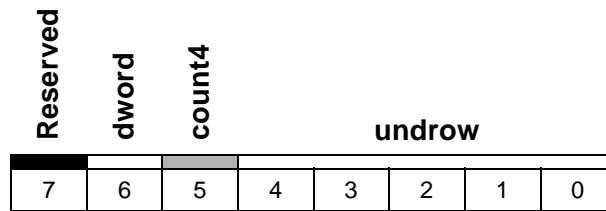
These bits are the eight LSBs of a 10-bit value that is used to offset the current line start address to the beginning of the next character row. Bits 8 and 9 are in register **CRTCEXT0**<5:4>. The value is the number of double words (**dword** (**CRTC14**<6>) = 1) or single words (**dword** = 0) in one line.



## Underline Location

CRTC14

**Index**            **crtcx** = 14h  
**Reset Value**    0000 0000 b



**undrow**            Horizontal row scan where the underline will occur. VGA.

**<4:0>**

These bits specify the horizontal row scan of a character row on which an underline occurs.

**count4<5>**        Count by 4. VGA.

- 0: Causes the memory address counter to be clocked as defined by the **count2** field (**CRTC17<3>**), 'count by two bits'.
- 1: Causes the memory address counter to be clocked with the character clock divided by four. The **count2** field, if set, will supercede **count4**, and the memory address counter will be clocked every two character clocks.

**dword<6>**         Double word mode. VGA.

- 0: Causes the memory addresses to be single word or byte addresses, as defined by the **wbmode** field (**CRTC17<6>**).
- 1: Causes the memory addresses to be double word addresses.

See the **CRTC17** register for the address table.

**Reserved**  
**<7>**

Writing has no effect. This field returns zero when read.

**CRTC15****Start Vertical Blank**

**Index**            **crtcx = 15h**  
**Reset Value**    **0000 0000 b**

**vblkstr**

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

**vblkstr  
<7:0>**

Start vertical blanking bits 7 to 0. VGA/MGA.

The vertical blank signal becomes active when the vertical line counter reaches the vertical blank start value (a 12-bit value). The lower eight bits are located here. Bit 8 is in **CRTC7**<3>, bit 9 is in **CRTC9**<5>, and bits 10 and 11 are in **CRTCEXT2**<4:3>.

**End Vertical Blank****CRTC16**

**Index**            **crtc<sub>x</sub>** = 16h  
**Reset Value**    0000 0000 b

**vblkend**

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

**vblkend**  
**<7:0>**            End vertical blanking. VGA/MGA.

The vertical blanking signal becomes inactive when, after being activated, the eight lower bits of the internal vertical line counter reach the end vertical blanking value.

## CRTC17

## CRTC Mode Control

Index **crtcx = 17h**  
 Reset Value 0000 0000 b

<b>crtcstN</b>	<b>wbmode</b>	<b>addwrap</b>	<b>Reserved</b>	<b>count2</b>	<b>hsyncsel</b>	<b>selrowscan</b>	<b>cms</b>
7	6	5	4	3	2	1	0

**cms<0>**

Compatibility mode support. VGA.

- 0: Select the row scan counter bit 0 to be output instead of memory counter address 13. See the tables below.
- 1: Select memory address 13 to be output. See the tables below.

## Memory Address Tables

*Legend:*

A: Memory address from the CRTC counter  
 RC: Row counter  
 MA: Memory address is sent to the memory controller

Double word access {**dword (CRTC14<6>), wbmode**} = 1X

	<b>{addwrap,selrowscan:cms}</b>			
<b>Output</b>	<b>X00</b>	<b>X01</b>	<b>X10</b>	<b>X11</b>
MA0	'0'	'0'	'0'	'0'
MA1	'0'	'0'	'0'	'0'
MA2	A0	A0	A0	A0
MA3	A1	A1	A1	A1
MA4	A2	A2	A2	A2
MA5	A3	A3	A3	A3
MA6	A4	A4	A4	A4
MA7	A5	A5	A5	A5
MA8	A6	A6	A6	A6
MA9	A7	A7	A7	A7
MA10	A8	A8	A8	A8
MA11	A9	A9	A9	A9
MA12	A10	A10	A10	A10
MA13	RC0	A11	RC0	A11
MA14	RC1	RC1	A12	A12
MA15	A13	A13	A13	A13

## CRTC Mode Control

... CRTC17 ...

*Word access {dword, wbmde} = 00*

	<i>{addwrap,selrowscan:cms}</i>							
<i>Output</i>	<i>000</i>	<i>001</i>	<i>010</i>	<i>011</i>	<i>100</i>	<i>101</i>	<i>110</i>	<i>111</i>
MA0	A13	A13	A13	A13	A15	A15	A15	A15
MA1	A0	A0	A0	A0	A0	A0	A0	A0
MA2	A1	A1	A1	A1	A1	A1	A1	A1
MA3	A2	A2	A2	A2	A2	A2	A2	A2
MA4	A3	A3	A3	A3	A3	A3	A3	A3
MA5	A4	A4	A4	A4	A4	A4	A4	A4
MA6	A5	A5	A5	A5	A5	A5	A5	A5
MA7	A6	A6	A6	A6	A6	A6	A6	A6
MA8	A7	A7	A7	A7	A7	A7	A7	A7
MA9	A8	A8	A8	A8	A8	A8	A8	A8
MA10	A9	A9	A9	A9	A9	A9	A9	A9
MA11	A10	A10	A10	A10	A10	A10	A10	A10
MA12	A11	A11	A11	A11	A11	A11	A11	A11
MA13	RC0	A12	RC0	A12	RC0	A12	RC0	A12
MA14	RC1	RC1	A13	A13	RC1	RC1	A13	A13
MA15	A14	A14	A14	A14	A14	A14	A14	A14

*Byte access {dword, wbmde} = 01*

	<i>{addwrap,selrowscan:cms}</i>			
<i>Output</i>	<i>X00</i>	<i>X01</i>	<i>X10</i>	<i>X11</i>
MA0	A0	A0	A0	A0
MA1	A1	A1	A1	A1
MA2	A2	A2	A2	A2
MA3	A3	A3	A3	A3
MA4	A4	A4	A4	A4
MA5	A5	A5	A5	A5
MA6	A6	A6	A6	A6
MA7	A7	A7	A7	A7
MA8	A8	A8	A8	A8
MA9	A9	A9	A9	A9
MA10	A10	A10	A10	A10
MA11	A11	A11	A11	A11
MA12	A12	A12	A12	A12
MA13	RC0	A13	RC0	A13
MA14	RC1	RC1	A14	A14
MA15	A15	A15	A15	A15

## ... CRTC17

## CRTC Mode Control

<b>selrowscan</b> <1>	Select row scan counter. VGA. <ul style="list-style-type: none"> <li>• 0: Select the row scan counter bit 1 to be output instead of memory counter address 14.</li> <li>• 1: Select memory address 14 to be output. See the tables in the <b>cms</b> field's description.</li> </ul>
<b>hsyncsel</b> <2>	Horizontal retrace select. VGA/MGA. <ul style="list-style-type: none"> <li>• 0: The vertical counter is clocked on every horizontal retrace.</li> <li>• 1: The vertical counter is clocked on every horizontal retrace divided by 2.</li> </ul> <p>This bit can be used to double the vertical resolution capability of the CRTC. All vertical timing parameters have a resolution of two lines in divided-by-two mode, including the scroll and line compare capability.</p>
<b>count2</b> <3>	Count by 2. VGA/MGA. <ul style="list-style-type: none"> <li>• 0: The <b>count4</b> field (<b>CRTC14</b>&lt;5&gt;) dictates if the character clock is divided by 4 (<b>count4</b> = 1) or by 1 (<b>count4</b> = 0).</li> <li>• 1: The memory address counter is clocked with the character clock divided by 2 (<b>count4</b> is 'don't care' in this case).</li> </ul>
<b>Reserved</b> <4>	Writing has no effect. Reading will give 0's.
<b>addwrap</b> <5>	Address wrap. VGA. <ul style="list-style-type: none"> <li>• 0: In word mode, select memory address counter bit 13 to be used as memory address bit 0. In byte mode, memory address counter bit 0 is used for memory address bit 0.</li> <li>• 1: In word mode, select memory address counter bit 15 to be used as memory address bit 0. In byte mode, memory address counter bit 0 is used for memory address bit 0. See the tables in the <b>cms</b> field's description.</li> </ul>
<b>wbmode</b> <6>	Word/byte mode. VGA. <ul style="list-style-type: none"> <li>• 0: When not in double word mode (<b>dword</b> (<b>CRTC14</b>&lt;6&gt;) = 0), this bit will rotate all memory addresses left by one position. Otherwise, addresses are not affected. In double word mode, this bit is 'don't care'. See the tables in the <b>cms</b> field's description.</li> <li>• 1: Select byte mode. The memory address counter bits are applied directly to the video memory.</li> </ul>
<b>crtcrstN</b> <7>	<b>CRTC</b> reset. VGA/MGA. <ul style="list-style-type: none"> <li>• 0: Force the horizontal and vertical sync to be inactive.</li> <li>• 1: Allow the horizontal and vertical sync to run.</li> </ul>

## Line Compare

CRTC18

**Index**            **crtcx = 18h**  
**Reset Value**    **0000 0000 b**

**linecomp**

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

**linecomp**  
**<7:0>**

Line compare. VGA/MGA.

When the vertical counter reaches the line compare value, the memory address counter is reset to '0'. This means that memory information located at 0 and up are displayed, rather than the memory information at the line compare.

This register is used to create a split screen:

- Screen A is located at memory start address (**CRTCC**, **CRTCD**) and up.
- Screen B is located at memory address 0 up to the **CRTCC**, **CRTCD** value.

The line compare value is an 11-bit value. Bits 7 to 0 reside here, bit 8 is in **CRTC7**<4>, bit 9 is in **CRTC9**<6>, and bit 10 is in **CRTCEXT2**<7>. The line compare unit is always a scan line that is independent of the **conv2t4** field (**CRTC9**<7>).

The line compare is also used to generate the vertical line interrupt.

**CRTC22****CPU Read Latch**

**Index**            **crtcx = 22h**  
**Reset Value**    **0000 0000 b**

**cpudata**

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

**cpudata  
<7:0>**

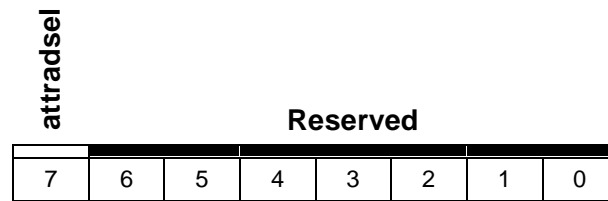
CPU data. VGA.

This register reads one of four 8-bit registers of the graphics controller CPU data latch. These latches are loaded when the CPU reads from display memory. The **rdmaps1** field (**GCTL4<1:0>**) determines which of the four planes is read in Read Mode 0. This register contains color compare data in Read Mode 1.



**Attributes Address/Data Select****CRTC24**

**Index**            **crtc<sub>x</sub>** = 24h  
**Reset Value**    0000 0000 b



**Reserved**  
**<6:0>**            Writing has no effect. This field returns all zeroes when read.

**attradssel**  
**<7>**                Attributes address/data select. VGA.

- 0: The attributes controller is ready to accept an address value.
- 1: The attributes controller is ready to accept a data value.

**CRTC26****Attributes Address**

**Index**            **crtc<sub>x</sub>** = 26h  
**Reset Value**    0000 0000 b

Reserved			pas		attrx		
7	6	5	4	3	2	1	0

**attrx<4:0>**    VGA attributes address  
**pas<5>**        VGA palette enable.  
**Reserved**  
**<7:6>**         Writing has no effect. This field returns all zeroes when read.

- See the **ATTR** register on page 4-78.

**CRTC Extension****CRTCEXT**

**Address** 03DEh (I/O), **mgabase1** + 1FDEh (MEM)  
**Attributes** R/W, BYTE/WORD, STATIC  
**Reset Value** nnnn nnnn 0000 0000 b

<b>crtcestd</b>								<b>Reserved</b>					<b>crtcestd</b>		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**crtcestd**  
**<2:0>**

CRTC extension index register.

A binary value that points to the CRTC Extension register where data is to be written or read when the **crtcestd** field is accessed.

<i>Register Name</i>	<i>Mnemonic</i>	<i>crtcestd address</i>
Address Generator Extensions	<b>CRTCEXT0</b>	00h
Horizontal Counter Extensions	<b>CRTCEXT1</b>	01h
Vertical Counter Extensions	<b>CRTCEXT2</b>	02h
Miscellaneous	<b>CRTCEXT3</b>	03h
Memory Page register	<b>CRTCEXT4</b>	04h
Horizontal Video Half Count	<b>CRTCEXT5</b>	05h
Reserved (1)	---	06h - 07h

(1) Writing to a reserved index has no effect; reading from a reserved index will give 0's.

**Reserved**  
**<7:3>**

Writing has no effect. This field returns all zeroes when read.

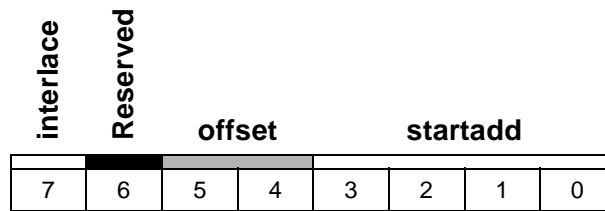
**crtcestd**  
**<15:8>**

CRTC extension data register.

Retrieves or writes the contents of the register pointed to by the **crtcestd** field.

**CRTCEXT0****Address Generator Extensions**

**Index**            **crtcextx = 00h**  
**Reset Value**    **0000 0000 b**



- startadd**  
**<3:0>**            Start address bits 19, 18, 17, and 16.  
These are the four most significant bits of the start address. See the **CRTCC** register on page 4-101.
- offset**  
**<5:4>**            Logical line width of the screen bits 9 and 8.  
These are the two most significant bits of the offset. See the **CRTC13** register on page 4-108.
- Reserved**  
**<6>**                Writing has no effect. This field returns zero when read.
- interlace**  
**<7>**                Interlace enable.  
Indicates if interlace mode is enabled.
- 0: Not in interlace mode.
  - 1: Interlace mode.

## Horizontal Counter Extensions

## CRTCEXT1

Index **crtcextx** = 01h  
 Reset Value 0000 0000 b

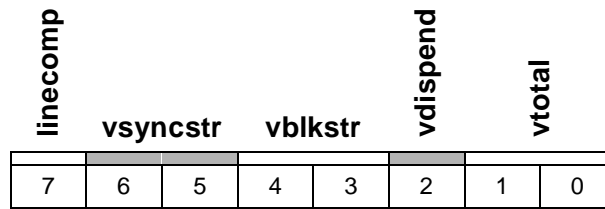
vrsten	hblkend	vsyncoff	hsyncoff	hrsten	hsyncstr	hblkstr	htotal
7	6	5	4	3	2	1	0

- htotal**  
<0> Horizontal total bit 8.  
This is the most significant bit of the **htotal** (horizontal total) register. See the **CRTC0** register on page 4-89.
- hblkstr**  
<1> Horizontal blanking start bit 8.  
This is the most significant bit of the **hblkstr** (horizontal blanking start) register. See the **CRTC2** register on page 4-91.
- hsyncstr**  
<2> Horizontal retrace start bit 8.  
This is the most significant bit of the **hsyncstr** (horizontal retrace start) register. See the **CRTC4** register on page 4-93.
- hrsten**  
<3> Horizontal reset enable.  
When at '1', the horizontal counter can be reset by the VIDRST pin.
- hsyncoff**  
<4> Horizontal sync off.  
  - 0: HSYNC runs freely.
  - 1: HSYNC is forced inactive.
- vsyncoff**  
<5> Vertical sync off.  
  - 0: VSYNC runs freely.
  - 1: VSYNC is forced inactive.
- hblkend**  
<6> End horizontal blanking bit 6. This bit is used only in MGA mode (**mgamode** = 1; see **CRTCEXT3**).  
Bit 6 of the End Horizontal Blanking value. See the **CRTC3** register on page 4-92.
- vrsten**  
<7> Vertical reset enable.  
When at '1', the vertical counter can be reset by the VIDRST pin.

## CRTCEXT2

## Vertical Counter Extensions

Index **crtcextx = 02h**  
 Reset Value 0000 0000 b



- vtotal**  
**<1:0>** Vertical total bits 11 and 10.  
 These are the two most significant bits of the **vtotal** (vertical total) register (the vertical total is then 12 bits wide). See the **CRTC6** register on page 4-95.
- vdispend**  
**<2>** Vertical display enable end bit 10.  
 This is the most significant bit of the **vdispend** (vertical display end) register (the vertical display enable end is then 11 bits wide). See the **CRTC12** register on page 4-107.
- vblkstr**  
**<4:3>** Vertical blanking start bits 11 and 10.  
 These are the two most significant bits of the **vblkstr** (vertical blanking start) register (the vertical blanking start is then 12 bits wide). See the **CRTC15** register on page 4-110.
- vsyncstr**  
**<6:5>** Vertical retrace start bits 11 and 10.  
 These are the two most significant bits of the **vsyncstr** (vertical retrace start) register (the vertical retrace start is then 12 bits wide). See the **CRTC10** register on page 4-105.
- linecomp**  
**<7>** Line compare bit 10.  
 This is the most significant bit of the **linecomp** (line compare) register (the line compare is then 11 bits wide). See the **CRTC18** register on page 4-115.

## Miscellaneous

## CRTCEXT3

**Index**            **crtcextx** = 03h  
**Reset Value**    0000 0000 b

<b>mgamode</b>	<b>csyncen</b>	<b>slow256</b>	<b>viddelay</b>	<b>scale</b>			
7	6	5	4	3	2	1	0

**scale<2:0>**    Dot clock scaling factor. Specifies the VCLK division factor in MGA mode.

<i>Scale</i>	<i>Division Factor</i>
000	/1
001	/2
010	/3
011	/4
100	Reserved
101	/6
110	Reserved
111	/8

**viddelay**  
**<4:3>**            Video delay.

Specifies the delay between the CRTIC signals and the delayed external signals. The number of delays to be added to the signal depends on the product's configuration:

<b>viddelay</b>	<i>Configuration</i>
00	4 MB board
01	2 MB board
1x	8 MB board

**slow256<5>**    256 color mode acceleration disable.

- 0: Direct frame buffer accesses are accelerated in VGA mode 13.
- 1: VGA Mode 13 direct frame buffer access acceleration is disabled. Unless otherwise specified, this bit should always be '0'.

**csyncen<6>**    Composite sync enable.

Generates a composite sync signal on the **VHSYNC/** pin.

- 0: Horizontal sync.
- 1: Composite sync (block sync).

**mgamode**  
<7>

MGA mode enable.

- 0: Select VGA compatibility mode. In this mode, VGA data to the RAMDAC is output on the DQ bus. The memory address counter clock will be selected by the **count2 (CRTC17<3>** and **count4 (CRTC14<5>**) bits. This mode should be used for all VGA modes up to mode 13, and for all Super VGA alpha modes. The VGA port of the RAMDAC should be selected. The load clock that is sent to the RAMDAC is **VCLK** (or VCLK/2 if mode 13 is selected). When **mga-mode** = '0', the full frame buffer aperture mapped to **mgabase2** is unusable.
- 1: Select MGA mode. In this mode, the graphics engine data is output on the DQ bus. The memory address counter is clocked with the VCLK pin divided by 2. This mode should be used for all Super VGA graphics modes and all accelerated graphics modes. The main port of the RAMDAC should be selected. The load clock that is sent to the RAMDAC is always VCLK.



## Memory Page

## CRTCEXT4

**Index**            **crtcextx** = 04h  
**Reset Value**    0000 0000 b

Res.		page					
7	6	5	4	3	2	1	0

**page**  
**<6:0>**

Page.

This register provides the extra bits required to address the full frame buffer through the VGA memory aperture in Power Graphic mode. This field must be programmed to zero in VGA mode. Up to 8 Mbytes of memory can be addressed. The **page** register can be used instead of or in conjunction with the MGA frame buffer aperture.

<b>GCTL6</b> <3:2>	<i>Bits used to address WRAM</i>	<i>Comment</i>
00	<b>CRTCEXT4</b> <6:1>, CPUA<16:0>	128K window
01	<b>CRTCEXT4</b> <6:0>, CPUA<15:0>	64K window
1X	Undefined	Window is too small

**Reserved**  
**<7>**

Writing has no effect. This field returns zero when read.

**CRTCEXT5****Horizontal Video Half Count**

**Index**            **crtcextx = 05h**  
**Reset Value**    **XXXX XXXX b**

**hvidmid**

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

**hvidmid**  
**<7:0>**

Horizontal video half count.

This register specifies the horizontal count at which the vertical counter should be clocked when in interlaced display in field 1. This register is only used in interlaced mode. The value to program is:

$$\frac{\text{Start Horizontal Retrace} + \text{End Horizontal Retrace} - \text{Horizontal Total}}{2} - 1$$

**DAC Status****DACSTAT**

**Address** 03C7h (I/O), **mgabase1** + 1FC7h (MEM)  
**Attributes** RO, BYTE, STATIC  
**Reset Value** 0000 0000 b

Reserved						dsts	
7	6	5	4	3	2	1	0

**dsts**  
**<1:0>**

This port returns the last access cycle to the palette.

- 00: Write palette cycle
- 11: Read palette cycle

Reads from the DAC write (3C8) or DAC status registers (3C7) do not interfere with read or write cycles, and they may take place at any time.

**Reserved**  
**<7:2>**

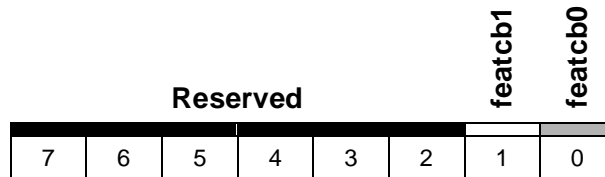
This field returns all zeroes when read. Data read at 03C7h will not be transmitted to the RAMDAC, and the contents of the **DACSTAT** register will be presented on the PCI bus. Writes to 03C7h will be transmitted to the RAMDAC.

**FEAT****Feature Control**

**Address** 03BAh (I/O), **mgabase1** + 1FBAh (MEM) Write monochrome  
 03DAh (I/O), **mgabase1** + 1FDAh (MEM) Write color  
 03CAh (I/O), 1FCAh (MEM) read

**Attributes** R/W, BYTE, STATIC

**Reset Value** 0000 0000 b



**featcb0<0>** Feature control bit 0. VGA. General read/write bit.

**featcb1<1>** Feature control bit 1. VGA. General read/write bit.

**Reserved <7:2>** Writing has no effect. This field returns all zeroes when read.

**Graphic Controller****GCTL**

**Address** 03CEh (I/O), **mgabase1** + 1FCEh (MEM)  
**Attributes** R/W, BYTE/WORD, STATIC  
**Reset Value** nnnn nnnn 0000 0000 b

gctld								Reserved				gctlx			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**gctlx** Graphic controller index register.

**<3:0>**

A binary value that points to the VGA graphic controller register where data is to be written or read when the **gctld** field is accessed.

<i>Register name</i>	<i>Mnemonic</i>	<i>gctlx address</i>
Set/Reset	<b>GCTL0</b>	00h
Enable Set/Reset	<b>GCTL1</b>	01h
Color Compare	<b>GCTL2</b>	02h
Data Rotate	<b>GCTL3</b>	03h
Read Map Select	<b>GCTL4</b>	04h
Graphic Mode	<b>GCTL5</b>	05h
Miscellaneous	<b>GCTL6</b>	06h
Color Don't Care	<b>GCTL7</b>	07h
Bit Mask	<b>GCTL8</b>	08h
Reserved (1)	---	09h - 0Fh

(1) Writing to a reserved index has no effect;  
reading from a reserved index will give 0's.

**Reserved** Writing has no effect. This field returns all zeroes when read.

**<7:4>**

**gctld** Graphic controller data register.

**<15:8>**

Retrieve or write the contents of the register pointed to by the **gctlx** field.

**GCTL0****Set/Reset**

**Index**            **gctlx = 00h**  
**Reset Value**    **0000 0000 b**

Reserved				setrst			
7	6	5	4	3	2	1	0

**setrst**  
**<3:0>**

Set/reset. VGA.

These bits allow setting or resetting byte values in the four video maps:

- 1: Set the byte, assuming the corresponding set/reset enable bit is '1'.
- 0: Reset the byte, assuming the corresponding set/reset enable bit is '0'.

This register is active when the graphics controller is in write mode 0 and enable set/reset is activated.

**Reserved**  
**<7:4>**

Writing has no effect. This field returns all zeroes when read.

## Enable Set/Reset

GCTL1

**Index**            **gctlx** = 01h  
**Reset Value**    0000 0000 b

Reserved				setrsten			
7	6	5	4	3	2	1	0

**setrsten**  
**<3:0>**

Enable set/reset planes 3 to 0. VGA.

When a set/reset plane is enabled (the corresponding bit is '1') and the write mode is 0 (**wrmode** (**GCTL5**<1:0>) = 00), the value written to all eight bits of that plane represents the contents of the set/reset register. Otherwise, the rotated CPU data is used.

This register has no effect when not in Write Mode 0.

**Reserved**  
**<7:4>**

Writing has no effect. This field returns all zeroes when read.

**GCTL2****Color Compare**

**Index**            **gctlx** = 02h  
**Reset Value**    0000 0000 b

Reserved				refcol			
7	6	5	4	3	2	1	0

**refcol**  
**<3:0>**

Reference color. VGA.

These bits represent a 4-bit color value to be compared. If the host processor sets Read Mode 1 (**rdmode** (**GCTL5**<3>) = 1), the data returned from the memory read will be a '1' in each bit position where the four planes equal the reference color value. Only the planes enabled by the **GCTL7** ('Color Don't Care'; [page 4-138](#)) register will be tested.

**Reserved**  
**<7:4>**

Writing has no effect. This field returns all zeroes when read.



## Data Rotate

## GCTL3

**Index**            **gctlx** = 03h  
**Reset Value**    0000 0000 b

Reserved			funsel		rot		
7	6	5	4	3	2	1	0

**rot**  
**<2:0>**

Data rotate count bits 2 to 0. VGA.

These bits represent a binary encoded value of the number of positions to right-rotate the host data before writing in Mode 0 (**wrmode** (**GCTL5**<1:0>) = 00).

The rotated data is also used as a mask together with the **GCTL8** ('Bit Mask', page 4-139) register to select which pixel is written.

**funsel**  
**<4:3>**

Function select. VGA.

Specifies one of four logical operations between the video memory data latches and any data (the source depends on the write mode).

<b>funsel</b>	<i>Function</i>
00	Source unmodified
01	Source AND latched data
10	Source OR latched data
11	Source XOR latched data

**Reserved**  
**<7:5>**

Writing has no effect. This field returns all zeroes when read.

**GCTL4****Read Map Select**

**Index**            **gctlx** = 04h  
**Reset Value**    0000 0000 b

Reserved					rdmapsl		
7	6	5	4	3	2	1	0

**rdmapsl**  
**<1:0>**

Read map select. VGA.

These bits represent a binary encoded value of the memory map number from which the host reads data when in Read Mode 0. This register has no effect on the color compare read mode (**rdmode** (**GCTL5**<3>) = 1).

**Reserved**  
**<7:2>**

Writing has no effect. This field returns all zeroes when read.

## Graphics Mode

## GCTL5

Index **gctlx** = 05h  
 Reset Value 0000 0000 b

Reserved	mode256	srintmd	gcoddevmd	rdmode	Reserved	wrmode
7	6	5	4	3	2	1 0

**wrmode**  
**<1:0>**

Write mode select. VGA.

These bits select the write mode:

- 00 In this mode, the host data is rotated and transferred through the set/reset mechanism to the input of the Boolean unit.
- 01 In this mode, the CPU latches are written directly into the frame buffer. The BLU is not used.
- 10 In this mode, host data bit n is replicated for every pixel of memory plane n, and this data is fed to the input of the BLU.
- 11 Each bit of the value contained in the **setrst** field (**GCTL0**<3:0>) is replicated to 8 bits of the corresponding map expanded. Rotated system data is ANDed with the **GCTL8** ('Bit Mask', page 4-139) register to give an 8-bit value which performs the same function as **GCTL8** in Modes 0 and 2.

**rdmode**  
**<3>**

Read mode select. VGA.

- 0: The host reads data from the memory plane selected by **GCTL4**, unless **chain4** (**SEQ4**<3>) equals 1 (in this case, the read map has no effect).
- 1: The host reads the result of the color comparison.

**gcoddevmd**  
**<4>**

Odd/Even mode select. VGA

- 0: The **GCTL4** (Read Map Select) register controls which plane the system reads data from.
- 1: Selects the odd/even addressing mode. It causes CPU address bit A0 to replace bit 0 of the read plane select register, thus allowing A0 to determine odd or even plane selection.

**srintmd**  
**<5>**

Shift register interleave mode. VGA.

- 0: Normal serialization.
- 1: The shift registers in the graphics controller format:
  - Serial data with odd-numbered bits from both maps in the odd-numbered map
  - Serial data with the even-numbered bits from both maps in the even-numbered maps.

**... GCTL5**

**Graphics Mode**

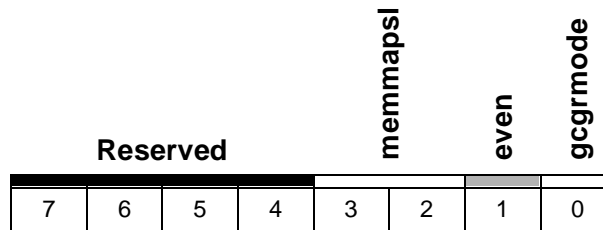
---

- mode256**<6> 256-color mode. VGA.
- 0: The loading of the shift registers is controlled by the **srintmd** field.
  - 1: The shift registers are loaded in a manner which supports 256-color mode.
- Reserved**  
<7> <2> Writing has no effect. These fields return all zeroes when read.

## Miscellaneous

## GCTL6

**Index** gctlx = 06h  
**Reset Value** 0000 0000 b



**gcgrmode**  
 <0> Graphics mode select. VGA.

- 0: Enables alpha mode, and the character generator addressing system is activated.
- 1: Enables graphics mode, and the character addressing system is not used.

**chainodd**  
 even<1> Odd/Even chain enable. VGA.

- 0: The A0 signal of the memory address bus is used during system memory addressing.
- 1: Allows A0 to be replaced by either the A16 signal of the system address (if **memmapsl** is '00'), or by the **hpgoddev** (**MISC**<5>, odd/even page select) field, described on [page 4-142](#)).

**memmapsl**  
 <3:2> Memory map select bits 1 and 0. VGA.

These bits select where the video memory is mapped, as shown below:

memmapsl	Address
00	A0000 - BFFFFh
01	A0000 - AFFFFh
10	B0000 - B7FFFh
11	B8000 - BFFFFh

**Reserved**  
 <7:4> Writing has no effect. This field returns all zeroes when read.

**GCTL7****Color Don't Care**

**Index**            **gctlx** = 07h  
**Reset Value**    0000 0000 b

Reserved				colcompen			
7	6	5	4	3	2	1	0

**colcompen**      Color enable comparison for planes 3 to 0. VGA.  
**<3:0>**            When any of these bits are set to '1', the associated plane is included in the color compare read cycle.

**Reserved**        Writing has no effect. This field returns all zeroes when read.  
**<7:4>**

**Bit Mask****GCTL8**

**Index**            **gctlx** = 08h  
**Reset Value**    0000 0000 b

**wrmask**

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

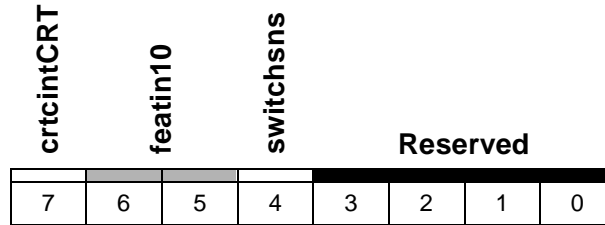
**wrmask  
<7:0>**

Data write mask for pixels 7 to 0. VGA.

If any bit in this register is set to '1', the corresponding bit in all planes may be altered by the selected write mode and system data. If any bit is set to '0', the corresponding bit in each plane will not change.

**INSTS0****Input Status**

**Address** 03C2h (I/O), **mgabase1** + 1FC2h (MEM) Read  
**Attributes** RO, BYTE, STATIC  
**Reset Value** N/A



**Reserved**  
**<3:0>** Writing has no effect. This field returns all zeroes when read.

**switchsns**  
**<4>** Switch sense bit. VGA.  
 Always read as 1. Writing has no effect.

**featin10**  
**<6:5>** Feature inputs 1 and 0. VGA.  
 Always read as '11'. Writing has no effect.

**crtcintCRT**  
**<7>** Interrupt.

- 0: Vertical retrace interrupt is cleared.
- 1: Vertical retrace interrupt is pending.



## Input Status 1

## INSTS1

<b>Address</b>	03BAh (I/O), <b>mgabase1</b> + 1FBAh (MEM) Read ( <b>MISC</b> <0> = 0, i.e mono-03DAh (I/O), <b>mgabase1</b> + 1FDAh (MEM) Read ( <b>MISC</b> <0> = 1, i.e color)
<b>Attributes</b>	RO, BYTE, DYNAMIC
<b>Reset Value</b>	N/A

<b>Reserved</b>		<b>diag</b>		<b>vretrace</b>	<b>Reserved</b>		<b>hretrace</b>
7	6	5	4	3	2	1	0

**hretrace**  
<0>

Display enable

- 0: Indicates an active display interval
- 1: Indicates an inactive display interval.

**vretrace**  
<3>

Vertical retrace.

- 0: Indicates that no vertical retrace interval is occurring.
- 1: Indicates a vertical retrace period.

**diag**  
<5:4>

Diagnostic.

The **diag** bits are selectively connected to two of the eight color outputs of the attribute controller. The **colplen** field (**ATTR12**<3:0>) determines which color outputs are used.

<b>vidstmx</b>		<b>diag</b>	
5	4	5	4
0	0	PD2	PD0
0	1	PD5	PD4
1	0	PD3	PD1
1	1	PD7	PD6

**Reserved**  
<7:6> <2:1>

Writing has no effect. These fields return all zeroes when read.

**MISC****Miscellaneous Output**

**Address** 03C2h (I/O), **mgabase1** + 1FC2h (MEM) Write 03CCh (I/O)  
**mgabase1** + 1FCCh (MEM) Read

**Attributes** R/W, BYTE, STATIC

**Reset Value** 0000 0000 b

<b>vsyncpol</b>	<b>hsyncpol</b>	<b>hpgoddev</b>	<b>videodis</b>	<b>clkssel</b>		<b>rammapen</b>	<b>ioaddsel</b>
7	6	5	4	3	2	1	0

**ioaddsel**  
**<0>** I/O address select. VGA.

- 0: The CRTC I/O addresses are mapped to 3BXh and the **STATUS** register is mapped to 03BAh for MDA emulation.
- 1: CRTC addresses are set to 03DXh and the **STATUS** register is set to 03DA for CGA emulation.

**rammapen**  
**<1>** Enable RAM. VGA.

- Logical '0': disable mapping of the frame buffer on the host bus.
- Logical '1': enable mapping of the frame buffer on the host bus.

**clkssel**  
**<3:2>** Clock selects. VGA/MGA.

These bits select the clock source that drives the hardware.

- 00: Select the 25.175 MHz clock.
- 01: Select the 28.322 Mhz clock.
- 1X: Reserved in VGA mode. Used to program and control the PLL.

**videodis**  
**<4>** Video disable. VGA This bit is reserved and read as '0'.

**hpgoddev**  
**<5>** Page bit for odd/even. VGA.

This bit selects between two 64K pages of memory when in odd/even mode.

- 0: Selects the low page of RAM.
- 1: Selects the high page of RAM.

## Miscellaneous Output

... MISC

**hsyncpol**  
<6>

Horizontal sync polarity. VGA/MGA.

- Logical '0': active high horizontal sync pulse.
- Logical '1': active low horizontal sync pulse.

The vertical and horizontal sync polarity informs the monitor of the number of lines per frame.

<i>VSYNC</i>	<i>HSYNC</i>	<i>Description</i>
+	+	768 lines per frame (marked as Reserved for IBM VGA)
-	+	400 lines per frame
+	-	350 lines per frame
-	-	480 lines per frame

**vsyncpol**  
<7>

Vertical sync polarity. VGA/MGA.

- Logical '0': active high vertical sync pulse
- Logical '1': active low vertical sync pulse

**SEQ****Sequencer**

**Address** 03C4h (I/O), **mgabase1** + 1FC4h (MEM)  
**Attributes** R/W, BYTE/WORD, STATIC  
**Reset Value** nnnn nnnn 0000 0000 b

<b>seqd</b>								<b>Reserved</b>				<b>seqx</b>			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**seqx**  
**<2:0>**

Sequencer index register.

A binary value that points to the VGA sequencer register where data is to be written or read when the **seqd** field is accessed.

<i>Register name</i>	<i>Mnemonic</i>	<i>seqx address</i>
Reset	<b>SEQ0</b>	00h
Clocking Mode	<b>SEQ1</b>	01h
Map Mask	<b>SEQ2</b>	02h
Character Map Select	<b>SEQ3</b>	03h
Memory Mode	<b>SEQ4</b>	04h
Reserved (1)	---	05h - 07h

(1) Writing to a reserved index has no effect; reading from a reserved index will give 0's.

**Reserved**  
**<7:3>**

Writing has no effect. This field returns all zeroes when read.

**seqd**  
**<15:8>**

Sequencer data register.

Retrieve or write the contents of the register that is pointed to by the **seqx** field.

**Reset****SEQ0**

**Index**            **seqx = 00h**  
**Reset Value**    **0000 0000 b**

Reserved						syncrst	asyncrst
7	6	5	4	3	2	1	0

**asyncrst**  
**<0>**

Asynchronous reset. VGA.

- 0: For the IBM VGA, this bit was used to clear and stop the sequencer asynchronously. For MGA, this bit can be read or written (for compatibility) but it does not stop the memory controller.
- 1: For the IBM VGA, this bit is used to remove the asynchronous reset.

**syncrst**  
**<1>**

Synchronous reset. VGA.

- 0: For the IBM VGA, this bit was used to clear and stop the sequencer at the end of a memory cycle. For MGA, this bit can be read or written (for compatibility), but it does not stop the memory controller. The MGA-2064W does not require that this bit be set to '0' when changing any VGA register bits.
- 1: For the IBM VGA, used to remove the synchronous reset.

**Reserved**  
**<7:2>**

Writing has no effect. This field returns all zeroes when read.

## SEQ1

## Clocking Mode

Index **seqx = 01h**  
 Reset Value 0000 0000 b

		<b>Reserved</b>		<b>scroff</b>	<b>shiffour</b>	<b>dotclkrt</b>	<b>shftldrt</b>		<b>Reserved</b>	<b>dotmode</b>
7	6	5	4	3	2	1	0			

**dotmode** 9/8 dot mode. VGA.

**<0>**

- 0: The sequencer generates a 9-dot character clock.
- 1: The sequencer generates an 8-dot character clock.

**shftldrt**

Shift/load rate. VGA.

**<2>**

- 0: The graphics controller shift registers are reloaded every character clock.
- 1: The graphics controller shift registers are reloaded every other character clock. This is used for word fetches.

**dotclkrt**

Dot clock rate. VGA.

**<3>**

- 0: The dot clock rate is the same as the video clock at the **VCLK** pin.
- 1: The dot clock rate is slowed to one-half the clock at the VCLK pin. The character clock and shift/load signals are also slowed to half their normal speed.

**shiffour**

Shift four. VGA.

**<4>**

- 0: The graphics controller shift registers are reloaded every character clock.
- 1: The graphics controller shift registers are reloaded every fourth character clock. This is used for 32-bit fetches.

**scroff**

Screen off. VGA/MGA.

**<5>**

- 0: Normal video operation.
- 1: Turns off the video, and maximum memory bandwidth is assigned to the system. The display is blanked, however all sync pulses are generated normally.

**Reserved**

Writing has no effect. These fields return all zeroes when read.

**<7:6> <1>**

**Map Mask****SEQ2**

**Index**            **seqx = 02h**  
**Reset Value**    **0000 0000 b**

Reserved				plwren			
7	6	5	4	3	2	1	0

**plwren**  
**<3:0>**

Map 3, 2, 1 and 0 write enable. VGA.

A '1' in any bit location will enable CPU writes to the corresponding video memory map. Simultaneous writes occur when more than one bit is '1'.

**Reserved**  
**<7:4>**

Writing has no effect. This field returns all zeroes when read.

## SEQ3

## Character Map Select

Index **seqx** = 03h  
 Reset Value 0000 0000 b

<b>Reserved</b>	<b>mapasel</b>	<b>mapbsel</b>	<b>mapasel</b>	<b>mapbsel</b>			
7	6	5	4	3	2	1	0

This register is reset by the reset pin (**PRST/**), or by the **asynrst** field of the **SEQ0** register.

**mapbsel**  
**<4,1:0>**

Map B select bits 2, 1, and 0. VGA.

These bits are used for alpha character generation when the character's attribute bit 3 is '0', according to the following table:

<b>mapbsel</b>	<b>Map#</b>	<b>Map location</b>
000	0	1st 8KB of Map 2
001	1	3rd 8KB of Map 2
010	2	5th 8KB of Map 2
011	3	7th 8KB of Map 2
100	4	2nd 8KB of Map 2
101	5	4th 8KB of Map 2
110	6	6th 8KB of Map 2
111	7	8th 8KB of Map 2

**mapasel**  
**<5,3:2>**

Map A select bits 2, 1, and 0. VGA.

These bits are used for alpha character generation when the character's attribute bit 3 is '1', according to the following table:

<b>mapasel</b>	<b>Map#</b>	<b>Map location</b>
000	0	1st 8KB of Map 2
001	1	3rd 8KB of Map 2
010	2	5th 8KB of Map 2
011	3	7th 8KB of Map 2
100	4	2nd 8KB of Map 2
101	5	4th 8KB of Map 2
110	6	6th 8KB of Map 2
111	7	8th 8KB of Map 2

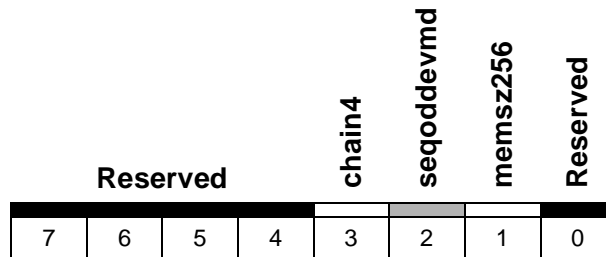
**Reserved**  
**<7:6>**

Writing has no effect. This field returns all zeroes when read.



**Memory Mode****SEQ4**

**Index**            **seqx = 04h**  
**Reset Value**    **0000 0000 b**



**memsz256**        256K memory size.  
**<1>**

- Set to '0' when 256K of memory is not installed. Address bits 14 and 15 are forced to '0'.
- Set to '1' when 256K of memory is installed. This bit should always be '1'.

**seqoddevmd**    Odd/Even mode. VGA.  
**<2>**

- 0: The CPU writes to Maps 0 and 2 at even addresses, and to Maps 1 and 3 at odd addresses.
- 1: The CPU writes to any map.

Note: In all cases, a map is written unless it has been disabled by the map mask register.

**chain4**            Chain four. VGA.  
**<3>**

- 0: The CPU accesses data sequentially within a memory map.
- 1: The two low-order bits A0 and A1 select the memory plane to be accessed by the system as shown below:

A<1:0>	Map selected
00	0
01	1
10	2
11	3

**Reserved**        Writing has no effect. These fields return all zeroes when read.  
**<7:4> <0>**





## ***Chapter 5: Programmer's Specification***

***This chapter includes:***

PCI Interface .....	5-2
WRAM Interface .....	5-11
Chip Configuration and Initialization .....	5-14
Direct Frame Buffer Access.....	5-18
Drawing in Power Graphic Mode .....	5-19
CRTC Programming .....	5-50
Interrupt Programming.....	5-58
DCI Support and Programming .....	5-59
Power Saving Features .....	5-60


## 5.1 PCI Interface

### 5.1.1 Direct Access Read Cache

Direct read accesses to the frame buffer (either by the MGA full frame buffer aperture or the VGA window) are cached by one 4-dword cache entry. After a hard or soft reset, the cache is emptied and the first direct read from the frame buffer fills the cache. Normally this read takes 10 pclk to complete. If the data of the following read is in the cache (as it normally should be three times out of four), the data phase of the access will be completed in 2 pclk.

The following situations will cause a cache flush, in order to maintain data coherency:

1. A write access to the frame buffer (**MGABASE2** or VGA frame buffer).
2. A write to the VGA registers (either I/O or memory).
3. Read accesses by the EPROM, RAMDAC, or external devices.
4. A VGA frame buffer read in VGA compatibility mode (**mgamode** = 0).
5. A hard or soft reset.

 Note: The cache is not flushed when the frame buffer configuration is modified (or when the drawing engine writes to a cached location). It is therefore the software's responsibility to invalidate the cache using one of the methods listed above whenever any bit that affects the frame buffer configuration or contents is written.

### 5.1.2 Big Endian Support

PCI may be used as an expansion bus for either little-endian or big-endian processors. The host-to-PCI bridge should be implemented to enforce address-invariance, as required by the *PCI Specification*. Address invariance means, for example, that when memory locations are accessed as bytes they return data in the same format. When this is done, however, non 8-bit data will appear to be 'byte-swapped'. Certain actions are then taken within the MGA-2064W to correct this situation.

The exact action that will be taken depends on the data size (the MGA-2064W must be aware of the data size when processing big-endian data). The data size depends on the location of the data (the specific memory space), and the pixel size (when the data is a pixel).

There are six distinct memory spaces:

1. Configuration space.
2. Boot space (EPROM).
3. I/O space.
4. Register space.
5. Frame buffer space.
6. ILOAD and IDUMP space.

#### Configuration space

Each register in the configuration space is 32 bits, and should be addressed using dword accesses. For these registers, no byte swapping is done, and bytes will appear in different positions, depending on the endian mode of the host processor. You must keep in mind that the MGA-2064W chip specification is written from the point of view of a little endian processor, and that the chip powers up in little endian mode.

#### Boot space (EPROM)

As with the configuration space, no special byte translation takes place. Proper byte organization can be achieved through correct EPROM programming. That is, data should be stored in big endian format for

big endian processors, and in little endian format for little endian processors.

### **I/O space**

Since I/O is only used on the MGA-2064W for VGA emulation, it should theoretically only be enabled on (little endian) x86 processors. However, it is still possible to use the I/O registers with other processor types because I/O accesses are considered to be 8-bit. In such a case, bytes should not be swapped anyway.

Byte swapping considerations aside, MGA-2064W I/O operations are mapped at fixed locations, which renders them incompatible with PCI's Plug and Play philosophy. This presents a second reason to avoid using the MGA-2064W I/O mapping on non x86 platforms.

### **Register space**

The majority of the data in the register space is 32 bits wide, with a few exceptions:

- The VGA compatibility section. Data in this section is 8 bits wide.
- The RAMDAC. Data in this section is 8 bits wide.
- External devices. In this case, the width of the data cannot be known in advance.

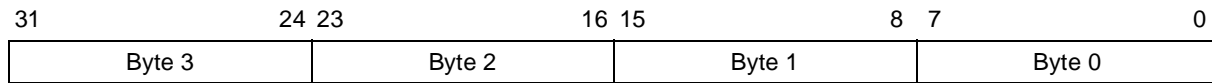
Byte swapping for big endian processors can be enabled in the register space by setting the **OPTION** configuration space register's **powerpc** bit to 1.

Setting the **powerpc** bit ensures that a 32-bit access by a big endian processor will load the correct data into a 32-bit register. In other words, when data is treated as 32 bit-quantities, it will appear in the identical way to both little and big endian processors. Note however that byte and word accesses will not return the same data on both little and big endian processors.

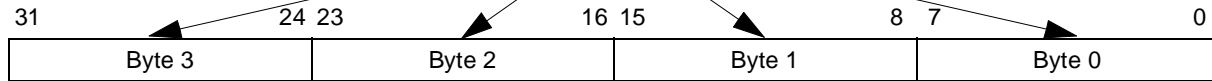
*In the register mapping tables in Chapter 3, all addresses are given for a little endian processor*

**powerpc = 1**

**PCI Bus**

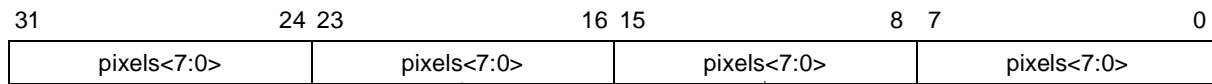


**Internal Register**

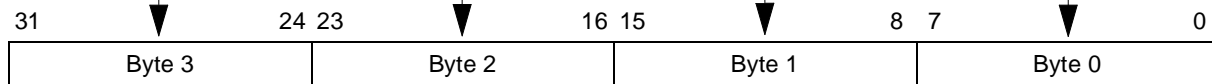


**powerpc = 0**

**PCI Bus**



**Internal Register**

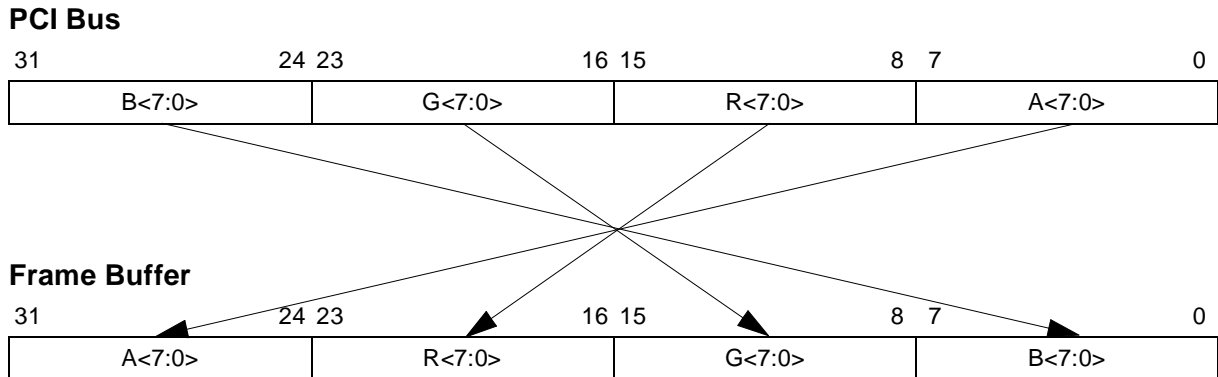


### Frame buffer space

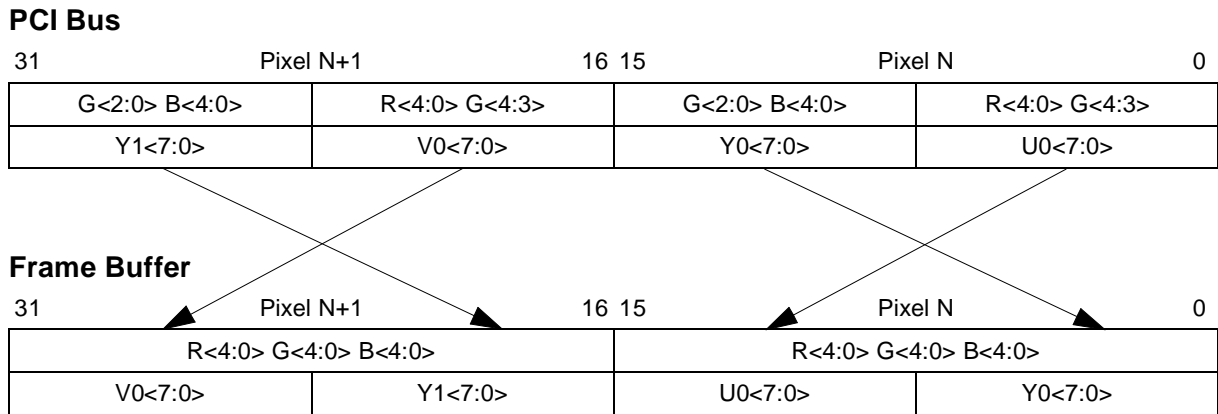
The frame buffer is organized in little endian format, and byte swapping depends on the size of the pixel. As usual, addresses are not modified.

Swapping mode is directed by the **dirDataSiz** field of the **OPMODE** host register. This field is used for direct access either through the VGA frame buffer window or the full memory aperture. The only exception is 24 bits/pixel mode, which is correctly supported only by little endian processors.

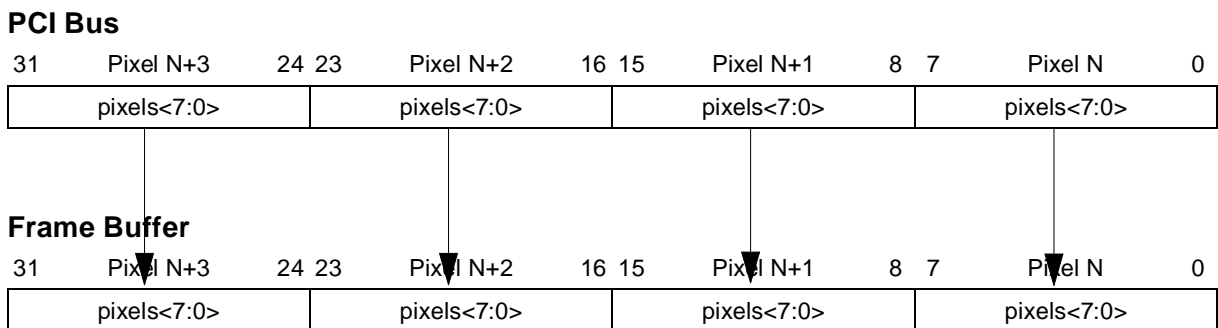
#### 32 bits/pixel, dirDataSiz = 10



#### 16 bits/pixel, dirDataSiz = 01



#### 8 bits/pixel, dirDataSiz = 00



## ILOAD & IDUMP space (DMAWIN)


Access to this space requires the same considerations as for the direct access frame buffer space (described previously), except that the **dmaDataSiz** field of the **OPMODE** register is used instead of **dirDataSiz** (for IDUMP or ILOAD operations in DMA BLIT WRITE mode). Other DMA modes - DMA General Purpose or DMA Vector Write - should set **dmaDataSiz** to '10'.

### 5.1.3 Host Pixel Format

There are several ways to access the frame buffer. The pixel format used by the host depends on the following:

- The current frame buffer's data format
- The access method
- The processor type (big endian or little endian)
- The control bits which select the type of byte swapping

The supported data formats are listed below, and are shown from the processor's perspective. The supported formats for direct frame buffer access, ILOAD, and IDUMP are explained in their respective sections of this chapter.

 Note: For big endian processors, these tables assume that the CPU-to-PCI bridge respects the *PCI Specification*, which states that byte address coherency must be preserved. This is the case for PREP systems and for Macintosh computers.

#### Pixel Format (from the processor's perspective)

##### 8-bit A

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Pixel 3								Pixel 2								Pixel 1								Pixel 0							
1	:								:								:								:							
2	:								:								:								:							
3	:								:								:								:							

##### 8-bit B

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Pixel 0								Pixel 1								Pixel 2								Pixel 3							
1	:								:								:								:							
2	:								:								:								:							
3	:								:								:								:							

##### 16-bit A

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Pixel 1																Pixel 0															
1	:																:															
2	:																:															
3	:																:															



**16-bit B**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Pixel 0																Pixel 1															
1	:																:															
2	:																:															
3	:																:															

**32-bit A**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Alpha Pixel 0								Red Pixel 0								Green Pixel 0								Blue Pixel 0							
1	Alpha Pixel 1								Red Pixel 1								Green Pixel 1								Blue Pixel 1							
2	Alpha Pixel 2								Red Pixel 2								Green Pixel 2								Blue Pixel 2							
3	:								:								:								:							

**32-bit B**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Alpha Pixel 0								Blue Pixel 0								Green Pixel 0								Red Pixel 0							
1	Alpha Pixel 1								Blue Pixel 1								Green Pixel 1								Red Pixel 1							
2	Alpha Pixel 2								Blue Pixel 2								Green Pixel 2								Red Pixel 2							
3	:								:								:								:							

**24-bit A**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Blue Pixel 1								Red Pixel 0								Green Pixel 0								Blue Pixel 0							
1	Green Pixel 2								Blue Pixel 2								Red Pixel 1								Green Pixel 1							
2	Red Pixel 3								Green Pixel 3								Blue Pixel 3								Red Pixel 2							
3	Blue Pixel 5								Red Pixel 4								Green Pixel 4								Blue Pixel 4							
4	:								:								:								:							

**24-bit B**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Red Pixel 1								Blue Pixel 0								Green Pixel 0								Red Pixel 0							
1	Green Pixel 2								Red Pixel 2								Blue Pixel 1								Green Pixel 1							
2	Blue Pixel 3								Green Pixel 3								Red Pixel 3								Blue Pixel 2							
3	Red Pixel 5								Blue Pixel 4								Green Pixel 4								Red Pixel 4							
4	:								:								:								:							

**YUV A**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	V0								Y1								U0								Y0							
1	V2								Y3								U2								Y2							
2	V4								Y5								U4								Y4							
3	:								:								:								:							

**YUV B**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Y1								V0								Y0								U0							
1	Y3								V2								Y2								U2							
2	Y5								V4								Y4								U4							
3	:								:								:								:							

**YUV C**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Y0								U0								Y1								V0							
1	Y2								U2								Y3								V2							
2	Y4								U4								Y5								V4							
3	:								:								:								:							

**YUV D**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	U0								Y0								V0								Y1							
1	U2								Y2								V2								Y3							
2	U4								Y4								V4								Y5							
3	:								:								:								:							

**MONO A**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	P31																P0															
1	P63																P32															
2	P95																P64															
3																	:															

P = 'pixel'

**MONO B**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	P24	...					P31	P16	...							P23	P8	...							P15	P0	...					P7
1	P56	...					P63	P48	...							P55	P40	...							P47	P32	...					P39
2	P88	...					P95	P80	...							P87	P72	...							P79	P64	...					P71
3					:					:					:					:												

**MONO C**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	P0																P31															
1	P32																P63															
2	P64																P95															
3																	:															

## 5.2 WRAM Interface

### 5.2.1 Frame Buffer Organization

The MGA-2064W supports up to three banks of memory. Banks 0 and 1 are 2 Mbytes each, while Bank 2 is 4 Mbytes. Using this configuration, it is possible to design a 2, 4, or 8 MByte product.

- Bank 0: 2 x 256Kx32 WRAM. Used as frame buffer memory.
- Bank 1: 2 x 256Kx32 WRAM. Used as frame buffer memory (optional).
- Bank 2: 4 x 256Kx32 WRAM. Used as frame buffer memory (optional).

There are three different frame buffer organizations, described below:

- VGA mode
- Power Graphic mode (non-interleave)
- Power Graphic mode (interleave)

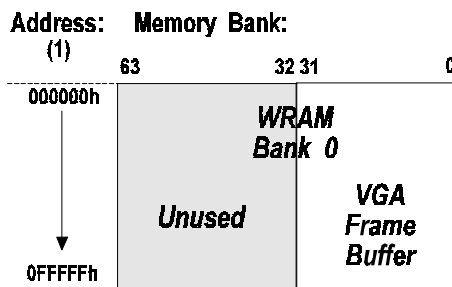
In Power Graphic mode, the resolution depends on the amount of available memory. The following table shows the memory requirements for each resolution and pixel depth. When 4M or 8M are required, interleave mode must be selected in order to have the proper bandwidth on the serial port.

#### Supported resolutions

<i>Resolution</i>	<i>8-bit</i>	<i>16-bit</i>	<i>24-bit</i>	<i>32-bit</i>
640 x 480	2M	2M	2M	2M
720 x 480	2M	2M	2M	2M
720 x 576	2M	2M	2M	2M
768 x 576	2M	2M	2M	2M
800 x 600	2M	2M	2M	2M
920 x 720	2M	2M	2M	4M
1024 x 768	2M	2M	4M	4M
1152 x 882	2M	2M	4M	4M
1280 x 1024	2M	4M	4M	8M
1600 x 1200	2M	4M	8M	-

#### 5.2.1.1 VGA mode

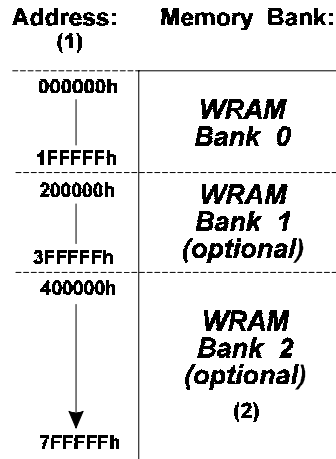
In VGA mode, the frame buffer can be up to 1M. In a 64-bit slice, byte line 0 is used as plane 0; byte line 1 is used as plane 1; byte line 2 is used as plane 2; byte line 3 is used as plane 3. Byte lines 4-7 are not used, and the contents of this memory are preserved. The contents of memory banks 1, 2, and 3 are also preserved.



(1) All addresses are hexadecimal byte addresses which correspond to pixel addresses in 8 bits/pixel mode.

### 5.2.1.2 Power Graphic Mode (non-interleave)

In this case, the frame buffer can be 2M, 4M, or 8M. This mode can be used with either a 32- or 64-bit RAMDAC.

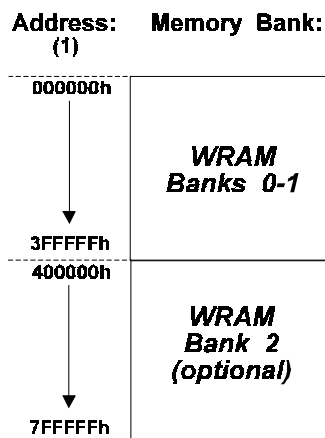


- (1) All addresses are hexadecimal byte addresses which correspond to pixel addresses in 8 bits/pixel mode.
- (2) This memory cannot be used for display, but must be used only for off-screen storage.

### 5.2.1.3 Power Graphic Mode (interleave)

The **interleave** bit in the **OPTION** register is used to configure the memory in interleave mode. A minimum of 4Mbytes (Banks 0 and 1) and a 64-bit RAMDAC are required for interleave mode.

#### Interleave organization



- (1) All addresses are hexadecimal byte addresses which correspond to pixel addresses in 8 bits/pixel mode.

## 5.2.2 Pixel Format

The slice is 64 bits long and is organized as follows. In all cases, the least significant bit is 0. The Alpha part of the color is the section of a pixel that is not used to drive the RAMDAC. Note that the data is always true color, but in 8 bit/pixel formats pseudo color can be used when shading is not used.

The 24 bit/pixel frame buffer organization is a special case wherein there are three different slice types. In this case, one pixel can be in two different slices.



## 5.3 Chip Configuration and Initialization

### 5.3.1 Reset

The MGA-2064W can be both hard and soft reset. Hard reset is achieved by activating the **PRST/** pin for more than 2 pclks. There is no need for the **PRST/** pin to be synchronous with any clock. Pulses shorter than 2 pclks will not trigger a hard reset.

- A hard reset will reset all chip registers to their reset values if such values exist. Refer to the individual register descriptions in [Chapter 4](#) to determine which bits are hard reset.
- All state machines are reset (possibly with termination of the current operation).
- FIFOs will be emptied, and the cache will be invalidated.
- A hard reset will activate the local bus reset (**EXTRST/**) in order to reset expansion devices when required. The **EXTRST/** signal is synchronous on **PCLK**, but not on **GCLK**.

The state of the straps are read and registered internally upon hard reset. A soft reset will not re-read the external straps or change the state of the **vgaioen** bit of the **OPTION** register. The state of the biosen internal strap cannot be read directly, but a test can be done to find its value. If the **ROMBASE** address can be written, then biosen = '1' otherwise biosen = '0'. The vgaBOOT strap is bit 23 of the **CLASS** register.

<i>Strap Name</i>	<i>Pins</i>	<i>Description</i>
biosen	MDQ<6>	Indicates whether a ROM is installed ('1') or not ('0'). The biosen strap also controls the <b>biosen</b> field of the <b>OPTION</b> register.
pid<4:0>	MDQ<4:0>	User-defined
vgaboot	MDQ<5>	Indicates whether the VGA I/O locations are decoded ('1') or not ('0') only if the <b>vgaioen</b> bit has not been written. The vgaBOOT strap also controls the <b>CLASS</b> register, setting the <b>class</b> field to 'Super VGA compatible controller' ('1') or to 'Other display controller' ('0').

A soft reset is performed by programming a '1' into bit 0 of the **RST** host register. Soft reset will be maintained until a '0' is programmed (see the **RST** register description on [page 4-61](#) for the details).

The soft reset should be interpreted as a drawing engine reset more than as a general soft reset. The video circuitry and the VGA registers, for example, are not affected by a soft reset. Only circuitry in the host section which affects the path to the drawing engine will be reset. Soft reset has no effect on the **EXTRST/** line.

### 5.3.2 Power Up Sequence

Aside from the PCI initialization, certain bits in the **OPTION** register must be set, according to the devices in the system that the chip is used in. These bits, shown in the following table, are vital to the correct behavior of the chip:

Name	Reset Value	Description
<b>eepromwt</b>	'0'	To be set to '1' if a FLASH ROM is used, and writes are to be done to the ROM.
<b>nogscale</b>	'0'	To be set to '1' if gclk does not need to be divided by 4 (internally).
<b>interleave</b>	'0'	To be set to '1' if there are 4 or 8 Mbytes of memory on board and a 64-bit RAMDAC is installed.
<b>powerpc</b>	'0'	To be set to '1' to support big endian processor accesses.
<b>rfhcnt</b>	'0000'	The refresh counter defines the rate of MGA memory refresh. For a typical 40 MHz system, a value of 9 would be programmed.
<b>vgaioen</b>	vgaboot strap	Takes the strap value on hard reset, but is also writable: '0': VGA I/O locations are not decoded '1': VGA I/O locations are decoded.

#### WRAM Reset Sequence

In order to properly initialize the WRAM, the following sequence must be respected at power-up. Note that Steps 7 to 17 must be performed every time the **softreset** field of the **RST** register is used.

- Step 1.** Initialize the clock generator to the proper gclk value.
- Step 2.** Program the graphic pre-scaler (the **OPTION** register's **nogscale** field).
- Step 3.** Place the drawing engine in MGA mode (set the **CRTCEXT3** register's **mgamode** field to '1').
- Step 4.** Set the **scroff** blanking bit (**SEQ1<5>**) to prevent transfer.
- Step 5.** Initialize the **CRTC** (See Section 5.6, 'CRTC Programming' on page 5-45).
- Step 6.** Program the refresh register (the **OPTION** register's **rfhcnt** field).
- Step 7.** Set the **softreset** bit of the **RST** register.
- Step 8.** Wait a minimum of 200 us.
- Step 9.** Clear the **softreset** bit.
- Step 10.** Wait a minimum of 200 us. to allow performance of more than eight refresh cycles.
- Step 11.** Wait for the vertical retrace.
- Step 12.** Enable the video.
- Step 13.** Wait for the next vertical retrace.
- Step 14.** Set the **memreset** bit of the **MACCESS** register.
- Step 15.** Wait 1 us.
- Step 16.** Program a solid rectangle fill in block mode, as specified in Section 5.5.4, 'Trapezoid / Rectangle Fill Programming' on page 5-21.
- Step 17.** Wait until the drawing engine is idle.
- Step 18.** From this point on, the WRAM is initialized, and the drawing engine can be accessed. The drawing engine can also be placed in VGA mode (**mgamode** = 0).



### 5.3.3 Operation Mode Selection

The MGA-2064W provides three display modes: text (VGA/SVGA); VGA graphics; SVGA graphics.

- The text display uses a multi-plane configuration in which a character, its attributes, and its font are stored in these separate memory planes. All text modes are either VGA-compatible or extensions of the VGA modes.
- The VGA graphics modes can operate in either multi-plane or packed-pixel modes, as is the case with standard VGA.
- The SVGA modes operate in packed-pixel mode - they enable use of the graphics engine. Also, SVGA modes use the serial port of the WRAM for display refresh. This results in very high performance, with high resolution and a greater number of pixel depths.

#### Mode switching

The BIOS follows the procedure below when switching between video modes:

1. Wait for the vertical retrace.
2. Disable the video by using the **scroff** blanking bit (**SEQ1**<5>).
3. Select the VGA or SVGA mode by programming the **mgamode** field of the **CRTCEXT3** register.
4. If a text mode or VGA graphic mode is selected, program the VGA-compatible register to initialize the appropriate mode.
5. Initialize the CRTC (see Section 5.6).
6. Initialize the RAMDAC and the video PLL for proper operation.
7. Initialize the frame buffer.
8. Wait for the vertical retrace.
9. Enable the video by using the **scroff** blanking bit.

Note: The majority of the registers required for initialization can be accessed via the I/O space. For registers that are not mapped through the I/O space, or if the I/O space is disabled, indirect addressing by means of the **MGA\_INDEX** and **MGA\_DATA** registers can be used. This would permit a real mode application to select the video mode, even if the **MGABASE1** aperture is above 1M.

## 5.4 Direct Frame Buffer Access

There are two memory apertures: the VGA memory aperture, and the **MGABASE2** memory aperture

#### VGA Mode

The **MGABASE2** memory aperture should not be used, due to constraints imposed by the frame buffer organization. The VGA memory aperture operates as a standard VGA memory aperture. Note also that in VGA mode only 1 Mbyte of the frame buffer is accessible. The **CRTCEXT4** register must be set to 0.

#### Power Graphic Mode

Both memory apertures can be used to access the frame buffer. The full frame buffer memory aperture provides access to the frame buffer without using any paging mechanism. The VGA memory aperture provides access to the frame buffer for real mode applications.

The **CRTCEXT4** register provides an extension to the page register in order to allow addressing of the complete frame buffer. Accesses to the frame buffer are concurrent with the drawing engine, so there is no requirement to synchronize the process which is performing direct frame buffer access with the process which is using the drawing engine. The MGA-2064W can perform data swapping for big endian processors (the data swapping mode is selected by the **OPMODE** register's **dirdatasiz**<1:0> field).

There are no plane write masks available during direct frame buffer accesses.

## 5.5 Drawing in Power Graphic Mode

This section explains how to program the MGA-2064W's registers to perform various graphics functions.

### 5.5.1 Overview

To understand how this programming guide works, please refer to the following explanations:

1. All registers are presented in a table that lists the register's name, its function, and any comment or alternate function.
2. The table for each *type* of object (for example, line with *depth*, *solid* line, *constant-shaded* trapezoid) is presented as a module in a third-level subsection numbered, for example, as 5.5.3.2.
3. The description of each *type* of object contains a representation of the **DWGCTL** register. The drawing control register illustration is repeated for each object *type* because it can vary widely, depending on the current graphics operation (refer to the **DWGCTL** description, which starts on page 4-40).

#### Legend for DWGCTL Illustrations:

- When a field **must be set to one of several possible values for the current operation**, it appears as plus signs (+), one for each bit in the field. The valid settings are listed underneath.
  - When a field **can be set to any of several possible valid values**, it appears as hash marks (#), one for each bit in the field. The values must still be valid for their associated operations.
  - When a field **must be set to a specific value** then that value appears.
4. You must program the registers listed in the 'Global Initialization (all operations)' section below *for all graphics operations*. Once this initialization has been performed, you can select the various objects and object *types* and program the registers for them accordingly.

### 5.5.2 Global Initialization (all operations)

You must initialize the following registers for all graphics operations:

<i>Register</i>	<i>Function</i>	<i>Comment / Alternate Function</i>
<b>PITCH</b>	Set pitch	Specify destination address linearization ( <b>iy</b> field)
<b>YDSTORG</b>	Determine screen origin	
<b>MACCESS</b>	Set pixel format (8, 16, 24, 32 bpp)	Some limitations apply
<b>CXBNDRY</b>	Left/right clipping limits	Can use <b>CXLEFT</b> and <b>CXRIGHT</b> instead
<b>YTOP</b>	Top clipping limit	
<b>YBOT</b>	Bottom clipping limit	
<b>PLNWT</b>	Plane write mask	
<b>ZORG</b>	Z origin position	Only required for depth operations

### 5.5.3 Line Programming

The following subsections list the registers that must be specifically programmed for solid lines, lines that use a linestyle, and lines that have a depth component. Remember to program the registers listed in section 5.5.2 and subsection 5.5.3.1 first. Also, *the last register you program must be accessed in the 1D00-1DFFh range in order to start the drawing engine.*

#### 5.5.3.1 Slope Initialization

##### Non auto-init lines

This type of line is initiated when the **DWGCTL** register's opcode field is set to either **LINE\_OPEN** or **LINE\_CLOSE**. A **LINE\_CLOSE** operation draws the last pixel of a line, while a **LINE\_OPEN** operation does not draw the last pixel. **LINE\_OPEN** is mainly used with polylines, where the final pixel of a given line is actually the starting pixel of the next line. This mechanism avoids having the same pixel written twice.


Register	Function	Comment / Alternate Function
<b>AR0</b>	$2b$ <sup>(1)</sup>	
<b>AR1</b>	Error term: $2b - a - sdy$	
<b>AR2</b>	Minor axis increment: $2b - 2a$	
<b>SGN</b>	Vector quadrant <sup>(2)</sup>	
<b>YDST</b>	The x start position	
<b>YDSTLEN</b>	The y start position and vector length	Can use <b>YDST</b> and <b>LEN</b> instead; <b>must</b> use <b>YDST</b> and <b>LEN</b> when destination address is linear (i.e.. <b>ylin</b> = 1, see <b>PITCH</b> )

<sup>(1)</sup> Definitions:  $a = \max(|dY|, |dX|)$ ,  $b = \min(|dY|, |dX|)$ .

<sup>(2)</sup> Sets major or minor axis and positive or negative direction for x and y.

##### Auto-init lines

This type of line is initiated when the **DWGCTL** register's opcode field is set to either **AUTOLINE\_OPEN** or **AUTOLINE\_CLOSE**. Auto-init vectors **cannot be used** when the destination addresses are linear (**ylin** = 1).

 Auto-init vectors are automatic lines whose major/minor axes and Bresenham parameters (these determine the exact pixels that a line will be composed of) do not have to be manually calculated by the user or provided by the host.

Register	Function	Comment / Alternate Function
<b>XYSTRT</b>	The x and y starting position	Can use <b>AR5</b> , <b>AR6</b> , <b>XDST</b> , and <b>YDST</b> instead
<b>XYEND</b>	The x and y starting position	Can use <b>AR0</b> and <b>AR2</b> instead

### 5.5.3.2 Solid Lines

#### DWGCTL:

Res.	transc	pattern	bltmod		Res.	trans				bop				Res.	shftzero	sgnzero	arzero	solid	zmode			linear	atype			opcode						
0	0	0	0	0	1	0	0	#	#	#	#	+	+	+	+	0	1	0	0	1	0	0	0	0	0	+	+	+	+	+	+	+

- **bop**: uses any Boolean operation if **atype** is RSTR; if **atype** is RPL, **bop** must be loaded with 0000, 0011, 1100, or 1111
- **atype**: can only be RPL or RSTR
- **opcode**: must be set to LINE\_OPEN, LINE\_CLOSE, AUTOLINE\_OPEN, or AUTOLINE\_CLOSE

Register	Function	Comment / Alternate Function
<b>FCOL</b>	Foreground color	

### 5.5.3.3 Lines That Use a Linestyle

**DWGCTL:**

Res.	transc	pattern	bltmod				Res.	trans				bop				Res.	shftzero	sgnzero	arzero	solid	zmode				linear	atype			opcode			
0	#	0	0	0	1	0	0	#	#	#	#	+	+	+	+	0	0	0	0	0	0	0	0	0	+	+	+	+	+	+	+	+

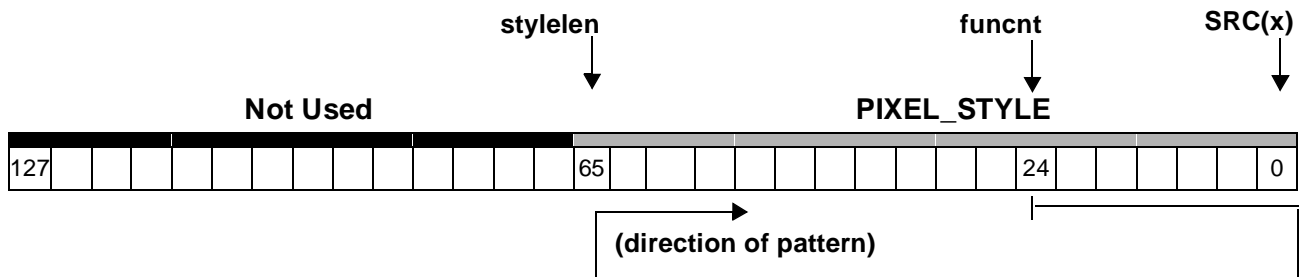
- **bop**: uses any Boolean operation if **atype** is RSTR; if **atype** is RPL, **bop** must be loaded with 0000, 0011, 1100, or 1111
- **atype**: can only be RPL or RSTR
- **opcode**: must be LINE\_OPEN, LINE\_CLOSE, AUTOLINE\_OPEN, or AUTOLINE\_CLOSE

Register	Function	Comment / Alternate Function
<b>SHIFT</b>	Linestyle length (stylelen), linestyle start point within the pattern (funcnt)	
<b>SRC0</b>	Linestyle pattern storage	
<b>SRC1</b>	Linestyle pattern storage	If <b>stylelen</b> is from 32-63
<b>SRC2</b>	Linestyle pattern storage	If <b>stylelen</b> is from 64-95
<b>SRC3</b>	Linestyle pattern storage	If <b>stylelen</b> is from 96-127
<b>BCOL</b>	Background color	If <b>transc</b> = 0
<b>FCOL</b>	Foreground color	

✍ To set up a linestyle, you must define the pattern you wish to use, and load it into the 128-bit source register (**SRC3-0**). Next, you must program **SHIFT** to indicate the length of your pattern minus 1 (**stylelen**). Finally, the **SHIFT** register's **funcnt** field is a count-down register with a wrap-around from zero to **stylelen**, which is used to indicate the point within the pattern at which you wish to start the linestyle. At the end of a line operation, **funcnt** points to the next value. For a polyline operation (LINE\_OPEN), the pixel style remains continuous with the next vector. With LINE\_CLOSE, the style does not increment with the last pixel.

**Linestyle illustration**

- SHIFT** : **stylelen** = 65, **funcnt** = 24
- SRC0** : **srcreg0** = PIXEL\_STYLE(31:0)
- SRC1** : **srcreg1** = PIXEL\_STYLE(63:32)
- SRC2** : **srcreg2** = PIXEL\_STYLE(65:64)



- The foreground color is written when the linestyle bit is '1'
- The background color is written when the linestyle bit is '0'


### 5.5.3.4 Lines with Depth

#### DWGCTL:

Res.	transc	pattern	bltmod				Res.	trans				bop				Res.	shftzero	sgnzero	arzero	solid	zmode			linear	atype			opcode			
0	0	0	0	0	1	0	0	#	#	#	#	1	1	0	0	0	0	0	0	#	#	#	0	+	+	+	+	+	+	+	

- **atype:** must be either ZI or I
- **opcode:** must be set to LINE\_OPEN, LINE\_CLOSE, AUTOLINE\_OPEN, or AUTOLINE\_CLOSE

Register	Function	Comment / Alternate Function
DR0	The z start position	Only if <b>zmode</b> <> NOZCMP or <b>atype</b> = ZI
DR2	The z major increment	Only if <b>zmode</b> <> NOZCMP or <b>atype</b> = ZI
DR3	The z diagonal increment	Only if <b>zmode</b> <> NOZCMP or <b>atype</b> = ZI
DR4	Red start position	
DR6	Red increment on major axis	
DR7	Red increment on diagonal axis	
DR8	Green start position	
DR10	Green increment on major axis	
DR11	Green increment on diagonal axis	
DR12	Blue start position	
DR14	Blue increment on major axis	
DR15	Blue increment on diagonal axis	


 Note that the **MACCESS** register's **pwidth** field must not be set to 24 bits per pixel (PW24) when drawing lines with depth.

## 5.5.4 Trapezoid / Rectangle Fill Programming

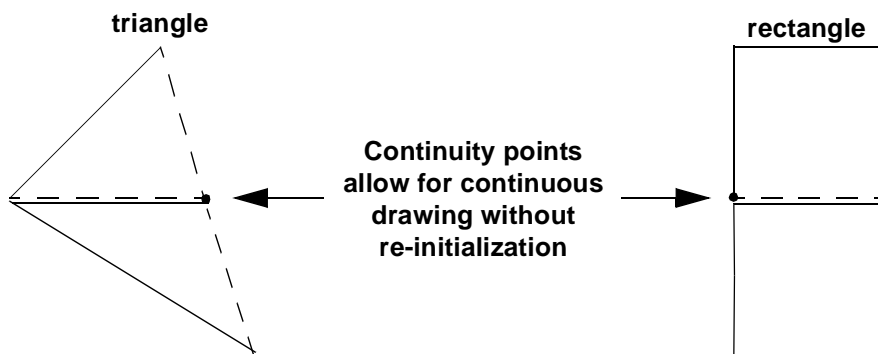
The following subsections list the registers that must be specifically programmed for constant and Gouraud shaded, patterned, and textured trapezoids, including rectangle and span line fills. Remember to program the registers listed in section 5.5.2 and in the tables in subsection 5.5.4.1 first. Also, *the last register you program must be accessed in the 1D00-1DFFh range in order to start the drawing engine.*

### 5.5.4.1 Slope Initialization

Trapezoids, rectangles, and span lines consist of a flat edge at the top and bottom, with programmable side edge positions at the left and right. When such a primitive is displayed, the pixels at the top and left edge are actually drawn as part of the object, while the bottom and right edges exist just beyond the object's extents. This is done so that when a primitive is completed, the common 'continuity points' that result allow a duplicate adjacent primitive to be drawn without the necessity of re-initializing all of the edges.

 Note that a primitive may have an edge of zero length, as in the case of a triangle (in this case, **FXRIGHT = FXLEFT**). You could draw a series of joined triangles by specifying the edges of the first triangle, then changing only one edge for each subsequent triangle.

- solid lines represent left, top edges
- dotted lines represent right, bottom edges



*Figure 5-1: Drawing Multiple Primitives*

## Trapezoids

The following registers must be initialized for trapezoid drawing:

<i>Register</i>	<i>Function</i>	<i>Comment / Alternate Function</i>
<b>AR0</b>	Left edge major axis increment: dYl yl_end - yl_start	
<b>AR1</b>	Left edge error term: errl ( <b>sdxl</b> == XL_NEG) ? dXl + dYl - 1 : - dXl	
<b>AR2</b>	Left edge minor axis increment: - dXl  - xl_end - xl_start	
<b>AR4</b>	Right edge error term: errr ( <b>sdxr</b> == XR_NEG) ? dXr + dYr - 1 : - dXr	
<b>AR5</b>	Right edge minor axis increment: - dXr  - xr_end - xr_start	
<b>AR6</b>	Right edge major axis increment: dYr yr_end - yr_start	
<b>SGN</b>	Vector quadrant	
<b>FXBNDRY</b>	Filled object x left and right coordinates	Can use <b>FXRIGHT</b> and <b>FXLEFT</b>
<b>YDSTLEN</b>	The y start position and number of lines	Can use <b>YDST</b> and <b>LEN</b> instead; <b>must</b> use <b>YDST</b> and <b>LEN</b> when destination address is linear (i.e.. <b>ylin</b> = 1, see <b>PITCH</b> )

## Rectangles and Span Lines

The following registers must be initialized for rectangle and span line drawing:

<i>Register</i>	<i>Function</i>	<i>Comment / Alternate Function</i>
<b>FXBNDRY</b>	Filled object x left and right coordinates	Can use <b>FXRIGHT</b> and <b>FXLEFT</b>
<b>YDSTLEN</b>	The y start position and number of lines	Can use <b>YDST</b> and <b>LEN</b> instead; <b>must</b> use <b>YDST</b> and <b>LEN</b> when destination address is linear (i.e.. <b>ylin</b> = 1, see <b>PITCH</b> )



### 5.5.4.2 Constant Shaded Trapezoids / Rectangle Fills

#### DWGCTL:

	Res.	transc	pattern	bltmod				Res.	trans				bop				Res.	shftzero	sgnzero	arzero	solid	zmode				linear	atype			opcode			
TRAP	0	0	0	0	0	0	0	0	+	+	+	+	+	+	+	+	0	1	0	0	1	0	0	0	0	0	+	+	+	0	1	0	0
RECT	0	0	0	0	0	0	0	0	+	+	+	+	+	+	+	+	0	1	1	1	1	0	0	0	0	0	+	+	+	0	1	0	0

- **trans**: if **atype** is BLOCK (block mode<sup>(1)</sup>), the transparency pattern is not supported - the value of **trans** must be '0000'
- **bop**: uses any Boolean operation if **atype** is RSTR; if **atype** is RPL, **bop** must be loaded with 0000, 0011, 1100, or 1111; if **atype** is BLOCK, **bop** must be loaded with 1100
- **atype**: can be RPL, RSTR, or BLOCK

Register	Function	Comment / Alternate Function
<b>FCOL</b>	Foreground color	

 Note that the **MACCESS** register's **pwidth** field can be set to 24 bits per pixel (PW24) with the following limitations:

- **atype** is either RPL or RSTR
- or*
- **forcol**<31:24>, **forcol**<23:16>, **forcol**<15:8>, and **forcol**<7:0> are set to the same value

<sup>(1)</sup> 'Block mode' refers to the high bandwidth block mode function of WRAM. It should be used whenever possible for the fastest performance, although certain restrictions apply (these are mentioned in comments regarding the BLOCK variable or 'block mode').

### 5.5.4.3 Patterned Trapezoids / Rectangle Fills

**DWGCTL:**

	Res.	transc	pattern	bltmod				Res.	trans				bop				Res.	shftzero	sgnzero	arzero	solid	zmode				linear	atype			opcode					
TRAP	0	#	0	0	0	0	0	0	+	+	+	+	+	+	+	+	+	0	#	0	0	0	0	0	0	0	0	0	+	+	+	0	1	0	0
RECT	0	#	0	0	0	0	0	0	+	+	+	+	+	+	+	+	0	#	1	1	0	0	0	0	0	0	0	+	+	+	0	1	0	0	

- **trans:** if **atype** is BLOCK, the transparency pattern is not supported - the value of **trans** must be '0000'
- **bop:** uses any Boolean operation if **atype** is RSTR; if **atype** is RPL, **bop** must be loaded with 0000, 0011, 1100, or 1111; if **atype** is BLOCK, **bop** must be loaded with 1100
- **atype:** Not required for little endian format. In Windows format, can be RPL, RSTR, or BLOCK

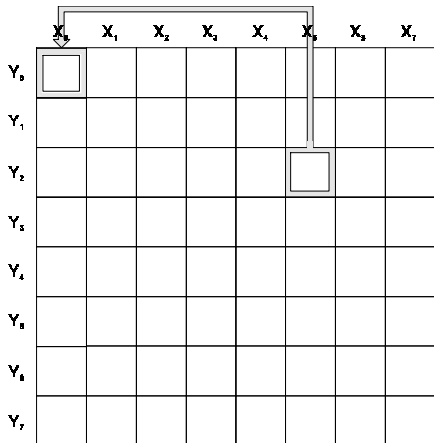
Register	Function	Comment / Alternate Function
PAT0	Pattern storage in Windows format	Use <b>SRC0, SRC1, SRC2, SRC3</b> for pattern storage in little endian format
PAT1		
SHIFT	Pattern origin offset	Only if <b>shftzero</b> = 0
BCOL	Background color	Only if <b>transc</b> = 0
FCOL	Foreground color	

Note that the **MACCESS** register's **pwidth** field can be set to 24 bits per pixel (PW24) with the following limitations:

- **atype** is either RPL or RSTR
- or
- **forcol**<31:24>, **forcol**<23:16>, **forcol**<15:8>, and **forcol**<7:0> are set to the same value, and **backcol**<31:24>, **backcol**<23:16>, **backcol**<15:8>, and **backcol**<7:0> are set to the same value.

#### Patterns and Pattern Offsets

Patterns can be comprised of one of two 8 x 8 pattern formats (Windows, or little endian). If required, you can offset the pattern origin for the frame buffer within the register (if no offset is required, program the **shftzero** bit to '1').

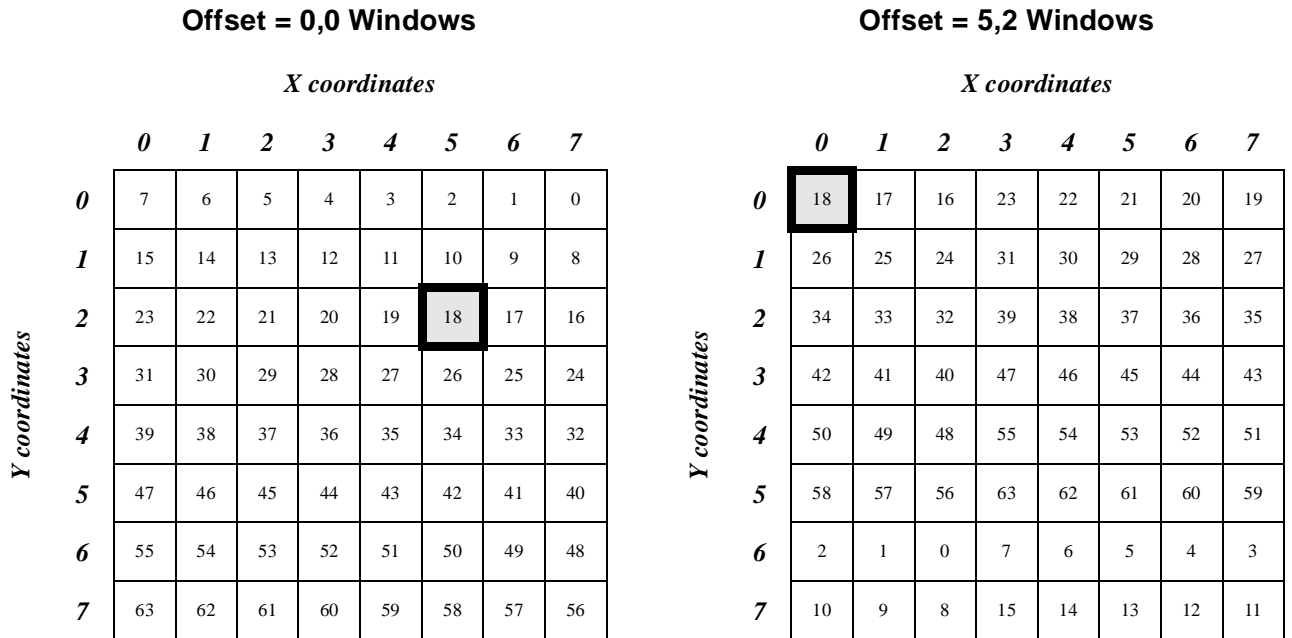


In the illustration on the left, the offset position is 5, 2. The corresponding register position's value is moved to the starting point of the pattern array. (This starting point is equivalent to an offset of 0,0.) Refer to the examples on the next page for more details.

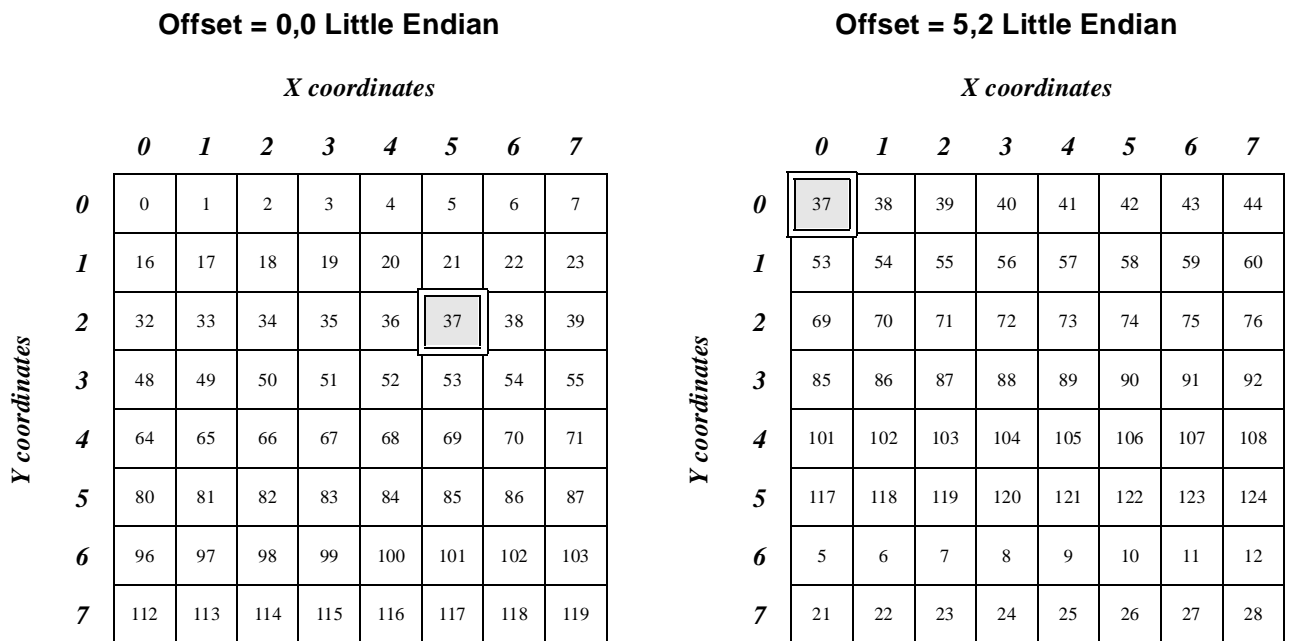
### Screen Representation

The examples below show how the data stored in the pattern registers is mapped into the frame buffer. The numbers inside the boxes represent the register bit positions that comprise the pattern.

- Windows format (used to drive Microsoft Windows) stores the pattern in the **PAT0** and **PAT1** registers. The following illustration shows the **PAT** register pattern usage for offsets of 0,0 and 5,2.



- Little endian format (for non-Windows systems) stores the pattern in the **SRC0**, **SRC1**, **SRC2**, and **SRC3** registers. In this case, the patterning for each line must be duplicated within the register (this simplifies software programming for hardware requirements). Depending on the offset, some pattern bits may come from the original pattern byte, while others may come from the associated duplicate byte. The following illustration shows the **SRC** register pattern usage for offsets of 0,0 and 5,2.



- For both formats, the foreground color is written when the pattern bit is '1'
- For both formats, the background color is written when the pattern bit is '0'

## 5.5.4.4 Gouraud Shaded Trapezoids / Rectangle Fills

## DWGCTL:

	Res.	transc	pattern	bltmod				Res.	trans				bop				Res.	shftzero	sgnzero	arzero	solid	zmode			linear	atype			opcode			
TRAP	0	0	0	0	0	0	0	0	#	#	#	#	1	1	0	0	0	1	0	0	0	#	#	#	0	+	+	+	0	1	0	0
RECT	0	0	0	0	0	0	0	0	#	#	#	#	1	1	0	0	0	1	1	1	0	#	#	#	0	+	+	+	0	1	0	0

- **atype**: must be either ZI or I

Register	Function	Comment / Alternate Function
<b>DR0</b>	The z start position	Only if <b>zmode</b> <> NOZCMP or <b>atype</b> = ZI
<b>DR2</b>	The z increment for x	Only if <b>zmode</b> <> NOZCMP or <b>atype</b> = ZI
<b>DR3</b>	The z increment for y	Only if <b>zmode</b> <> NOZCMP or <b>atype</b> = ZI
<b>DR4</b>	Red start position	
<b>DR6</b>	Red increment on x axis	
<b>DR7</b>	Red increment on y axis	
<b>DR8</b>	Green start position	
<b>DR10</b>	Green increment on x axis	
<b>DR11</b>	Green increment on y axis	
<b>DR12</b>	Blue start position	
<b>DR14</b>	Blue increment on x axis	
<b>DR15</b>	Blue increment on y axis	
<b>FCOL</b>	Alpha value	Only if <b>pwidth</b> = 32, or <b>pwidth</b> = 16 and <b>dit555</b> = 1.

- ✍ Note that the **MACCESS** register's **pwidth** field must not be set to 24 bits per pixel (PW24) when drawing Gouraud shaded trapezoids.

## 5.5.4.5 Textured Trapezoids / Rectangle Fills

## DWGCTL:

	Res.	transc	pattern	bltmod				Res.	trans				bop				Res.	shftzero	sgnzero	arzero	solid	zmode			linear	atype			opcode			
TRAP	0	0	0	+	+	+	+	0	#	#	#	#	1	1	0	0	0	1	0	0	0	#	#	#	0	+	+	+	0	1	0	1
RECT	0	0	0	+	+	+	+	0	#	#	#	#	1	1	0	0	0	1	1	1	0	#	#	#	0	+	+	+	0	1	0	1

- **bltmod**: must be one of the following: BU32BGR, BU32RGB, BU24BGR, or BU24RGB
- **atype**: must be either ZI or I

Register	Function	Comment / Alternate Function
<b>OPMODE</b>	Select DMA BLIT Write	
<b>DR0</b>	The z start position	Only if zmode $\neq$ NOZCMP or atype = ZI
<b>DR2</b>	The z increment for x	Only if zmode $\neq$ NOZCMP or atype = ZI
<b>DR3</b>	The z increment for y	Only if zmode $\neq$ NOZCMP or atype = ZI
<b>FCOL</b>	Alpha value	Only if <b>pwidth</b> = 32, or <b>pwidth</b> = 16 and <b>dit555</b> = 1.

- ✍ Note that the **MACCESS** register's **pwidth** field must not be set to 24 bits per pixel (PW24) when drawing Gouraud shaded trapezoids.
- ✍ **Note: It is important to transfer the exact number of pixels** expected by the drawing engine, since the drawing engine will not end the current operation until all pixels have been received. A deadlock will result if the host transfers *fewer pixels* than expected to the drawing engine (the software assumes the transfer is completed, but meanwhile the drawing engine is waiting for additional data). On the other hand, if the host transfers *more pixels* than expected, the extra pixels will be interpreted by the drawing engine as register accesses.

## 5.5.5 Bitblt Programming

The following subsections list the registers that must be specifically programmed for Bitblt operations. Remember to program the registers listed in section 5.5.2 and subsection 5.5.5.1 first. Also, *the last register you program must be accessed in the 1D00-1DFFh range in order to start the drawing engine.*

### 5.5.5.1 Address Initialization

#### XY Source Addresses

<i>Register</i>	<i>Function</i>	<i>Comment / Alternate Function</i>
<b>AR0</b>	Source end address	The last pixel of the first line
<b>AR3</b>	Source start address	
<b>AR5</b>	Source y increment	
<b>FXBNDRY</b>	Destination boundary (left and right)	Can use <b>FXRIGHT</b> and <b>FXLEFT</b>
<b>YDSTLEN</b>	The y start position and number of lines	Can use <b>YDST</b> and <b>LEN</b> instead

#### Linear Source Addresses

<i>Register</i>	<i>Function</i>	<i>Comment / Alternate Function</i>
<b>AR0</b>	Source end address	The last pixel of the source
<b>AR3</b>	Source start address	
<b>FXBNDRY</b>	Destination boundary (left and right)	Can use <b>FXRIGHT</b> and <b>FXLEFT</b>
<b>YDSTLEN</b>	The y start position and number of lines	Must use <b>YDST</b> and <b>LEN</b> when destination address is linear (i.e., <b>ylin</b> = 1, see <b>PITCH</b> )

AR3 comprises 18 bits, so a maximum of 256 Kpixels can be blitted.

#### Fast Bitblt and Patterning Operations

<i>Register</i>	<i>Function</i>	<i>Comment / Alternate Function</i>
<b>FXBNDRY</b>	Destination boundary (left and right)	Can use <b>fxright</b> and <b>fxleft</b>
<b>YDSTLEN</b>	The y start position and number of lines	Can use <b>YDST</b> and <b>LEN</b> instead; <b>must</b> use <b>YDST</b> and <b>LEN</b> when destination address is linear (i.e., <b>ylin</b> = 1, see <b>PITCH</b> )

## 5.5.6 Two-operand Bitblts

### DWGCTL:

	Res.	transc	pattern	bltmod			Res.	trans				bop				Res.	shftzero	sgnzero	arzero	solid	zmode			linear	atype			opcode					
XY	0	0	0	0	0	1	0	0	#	#	#	#	+	+	+	+	0	1	+	0	0	0	0	0	0	0	+	+	+	1	0	0	0
LIN.	0	0	0	0	1	1	1	0	#	#	#	#	+	+	+	+	0	1	1	0	0	0	0	0	0	1	+	+	+	1	0	0	0

- **bop**: uses any Boolean operation if **atype** is RSTR; if **atype** is RPL, **bop** must be loaded with 0000, 0011, 1100, or 1111
- **atype**: must be either RPL or RSTR

Register	Function	Comment / Alternate Function
<b>SGN</b>	Vector quadrant <sup>(1)</sup>	Only needs to be set when <b>sgnzero</b> = '0'

<sup>(1)</sup> Sets major or minor axis and positive or negative direction for x and y.

### 5.5.6.1 Two-operand Fast Bitblts

Register	Function	Comment / Alternate Function
<b>DWGCTL</b>	040A600C	
<b>AR0</b>	Source end address	The last pixel of the first line
<b>AR3</b>	Source start address	
<b>AR5</b>	Source y increment	

 When programming the **AR0**, **AR3**, and **FXBNDRY** registers, the destination must be aligned with the source, according to [Table 5-1](#):

*Table 5-1: Fast Bitblt Alignment Constraints*

interleave	pwidth	Alignment Constraint
0	PW8	$(\text{SRC\_START\_ADDRESS mod } 64) == ((\text{DST\_LEFT\_BND} + \text{Y\_LINEAR} + \text{yorg}) \text{ mod } 64)$
	PW16	$(\text{SRC\_START\_ADDRESS mod } 32) == (\text{DST\_LEFT\_BND mod } 32)$
	PW24	$(\text{SRC\_START\_ADDRESS mod } 64) == ((\text{DST\_LEFT\_BND} + \text{Y\_LINEAR} + \text{yorg}) \text{ mod } 64)$
	PW32	$(\text{SRC\_START\_ADDRESS mod } 16) == (\text{DST\_LEFT\_BND mod } 16)$
1	PW8	$(\text{SRC\_START\_ADDRESS mod } 128) == ((\text{DST\_LEFT\_BND} + \text{Y\_LINEAR} + \text{yorg}) \text{ mod } 128)$
	PW16	$(\text{SRC\_START\_ADDRESS mod } 64) == ((\text{DST\_LEFT\_BND} + \text{Y\_LINEAR} + \text{yorg}) \text{ mod } 64)$
	PW24	$(\text{SRC\_START\_ADDRESS mod } 128) == ((\text{DST\_LEFT\_BND} + \text{Y\_LINEAR} + \text{yorg}) \text{ mod } 128)$
	PW32	$(\text{SRC\_START\_ADDRESS mod } 32) == (\text{DST\_LEFT\_BND mod } 32)$



### 5.5.6.2 Color Patterning 8 x 8

#### DWGCTL:

Res.	transc	pattern	bltmod				Res.	trans				bop				Res.	shftzero	sgnzero	arzero	solid	zmode				linear	atype			opcode			
0	0	1	0	0	1	0	0	#	#	#	#	+	+	+	+	0	1	1	0	0	0	0	0	0	+	+	+	1	0	0	0	

- **bop**: uses any Boolean operation if **atype** is RSTR; if **atype** is RPL, **bop** must be loaded with 0000, 0011, 1100, or 1111
- **atype**: can be RPL or RSTR

Register	Function	Comment / Alternate Function
<b>AR0</b>	<b>AR0</b> <18:3> = <b>AR3</b> <18:3> When <b>pwidth</b> = PW8, <b>AR0</b> <2:0> = <b>AR3</b> <2:0> + 02h; when <b>pwidth</b> = PW16, <b>AR0</b> <2:0> = <b>AR3</b> <2:0> + 04h; when <b>pwidth</b> = PW32, <b>AR0</b> <2:0> = <b>AR3</b> <2:0> + 06h; when <b>pwidth</b> = PW24, <b>AR0</b> <2:0> = <b>AR3</b> <2:0> + 07h.	
<b>AR3</b>	Pattern address + x offset + (y offset * 32)	
<b>AR5</b>	32	

✍ The **AR3** register performs a dual function: it sets the pattern’s address, and it is also used to determine how the pattern will be pinned in the destination. Refer to ‘Patterns and Pattern Offsets’ on page 5-24, since color patterning is performed in a similar manner to monochrome patterning (except that the **SHIFT** register is not used for pinning).

- ✍ 8, 16, 32 bit/pixel pattern storage hardware restrictions:
  - The first pixel of the pattern must be stored at a pixel address module 256 + 0, 8, 16, or 24.
  - Each line of 8 pixels is stored continuously in memory for each pattern, but there must be a difference of 32 in the pixel address between each line of the pattern. To do this efficiently, four patterns should be stored in memory in an interleaved manner, in a block of 4 x 8 x 8 pixel locations. The following table illustrates such a pattern storage (the numbers in the table represent the pixel addresses, modulo 256):

		Pattern 0								Pattern 1								Pattern 2								Pattern 3							
Pixels:		0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
Lines:	0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	1	32	.	.	.	.	.	.	39	40	.	.	.	.	.	.	47	48	.	.	.	.	.	.	55	56	.	.	.	.	.	63	
	2	64	.	.	.	.	.	.	71	72	.	.	.	.	.	.	79	80	.	.	.	.	.	.	87	88	.	.	.	.	.	95	
	3	96	.	.	.	.	.	.	103	104	.	.	.	.	.	.	111	112	.	.	.	.	.	.	119	120	.	.	.	.	.	127	
	4	128	.	.	.	.	.	.	135	136	.	.	.	.	.	.	143	144	.	.	.	.	.	.	151	152	.	.	.	.	.	159	
	5	160	.	.	.	.	.	.	167	168	.	.	.	.	.	.	175	176	.	.	.	.	.	.	183	184	.	.	.	.	.	191	
	6	192	.	.	.	.	.	.	199	200	.	.	.	.	.	.	207	208	.	.	.	.	.	.	215	216	.	.	.	.	.	223	
	7	224	.	.	.	.	.	.	231	232	.	.	.	.	.	.	239	240	.	.	.	.	.	.	247	248	.	.	.	.	.	255	

- Pattern 3 is not available when the **MACCESS** register’s **pwidth** field is PW16 or PW32

 24 bit/pixel pattern storage hardware restrictions:

- The first pixel of the pattern must be stored at a pixel address modulo 256 + 0, or 16.
- Each line of 8 pixels is stored continuously in memory for each pattern, but there must be a difference of 32 in the pixel address between each line of the pattern. To do this efficiently, two patterns should be stored in memory in an interleaved manner, in a block of 2 x 16 x 8 pixel locations. The following table illustrates such a pattern storage (the numbers in the table represent the pixel addresses, modulo 256):

		<i>Pattern 0</i>																<i>Pattern 1</i>															
Pixels:		0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
Lines:	0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	1	32	.	.	.	.	.	.	.	.	.	.	.	.	.	.	47	48	.	.	.	.	.	.	.	.	.	.	.	.	.	.	63
	2	64	.	.	.	.	.	.	.	.	.	.	.	.	.	.	79	80	.	.	.	.	.	.	.	.	.	.	.	.	.	.	95
	3	96	.	.	.	.	.	.	.	.	.	.	.	.	.	.	111	112	.	.	.	.	.	.	.	.	.	.	.	.	.	.	127
	4	128	.	.	.	.	.	.	.	.	.	.	.	.	.	.	143	144	.	.	.	.	.	.	.	.	.	.	.	.	.	.	159
	5	160	.	.	.	.	.	.	.	.	.	.	.	.	.	.	175	176	.	.	.	.	.	.	.	.	.	.	.	.	.	.	191
	6	192	.	.	.	.	.	.	.	.	.	.	.	.	.	.	207	208	.	.	.	.	.	.	.	.	.	.	.	.	.	.	223
	7	224	.	.	.	.	.	.	.	.	.	.	.	.	.	.	239	240	.	.	.	.	.	.	.	.	.	.	.	.	.	.	255


### 5.5.6.3 BitBlts With Expansion (Character Drawing) 1 bpp

**DWGCTL:**

Res.	transc	pattern	bltmod				Res.	trans				bop				Res.	shftzero	sgnzero	arzero	solid	zmode				linear	atype			opcode			
0	#	0	0	0	0	0	0	+	+	+	+	+	+	+	+	+	0	1	1	0	0	0	0	0	#	+	+	+	1	0	0	0

- **trans:** if **atype** is BLOCK, the transparency pattern is not supported - the value of **trans** must be '0000'
- **bop:** uses any Boolean operation if **atype** is RSTR; if **atype** is RPL, **bop** must be loaded with 0000, 0011, 1100, or 1111; if **atype** is BLOCK, must be loaded with 1100
- **atype:** can be RPL, RSTR, or BLOCK

Register	Function	Comment / Alternate Function
<b>BCOL</b>	Background color	Only when transc = '0'
<b>FCOL</b>	Foreground color	

-  Note that the **MACCESS** register's **pwidth** field can be set to 24 bits per pixel (PW24) with the following limitations:
- **atype** is either RPL or RSTR
  - or*
  - **forcol**<31:24>, **forcol**<23:16>, **forcol**<15:8>, and **forcol**<7:0> are set to the same value, and **backcol**<31:24>, **backcol**<23:16>, **backcol**<15:8>, and **backcol**<7:0> are set to the same value.


### 5.5.6.4 BitBlts With Expansion (Character Drawing) 1 bpp Planar

**DWGCTL:**

Res.	transc	pattern	bltmod				Res.	trans				bop				Res.	shftzero	sgnzero	arzero	solid	zmode				linear	atype			opcode			
0	#	0	0	0	0	1	0	#	#	#	#	+	+	+	+	0	0	1	0	0	0	0	0	#	+	+	+	1	0	0	0	

- **bop:** uses any Boolean operation if **atype** is RSTR; if **atype** is RPL, **bop** must be loaded with 0000, 0011, 1100, or 1111
- **atype:** can be either RPL or RSTR

Register	Function	Comment / Alternate Function
<b>SHIFT</b>	Plane selection	
<b>BCOL</b>	Background color	Only when <b>transc</b> = '0'
<b>FCOL</b>	Foreground color	

 **MACCESS:** note that planar bitblts are not supported with 24 bits/pixel (PW24).

## 5.5.7 ILOAD Programming

The following subsections list the registers that must be specifically programmed for ILOAD (image load: Host -> VRAM) operations. You must take the following steps:

- Step 1.** Initialize the registers. Remember to program the registers listed in section 5.5.2 and subsection 5.5.7.1. Depending on the type of operation you wish to perform, you must also program the registers in subsection 5.5.7.2 or subsection 5.5.7.3.
- Step 2.** The last register you program must be accessed in the 1D00-1DFFh range in order to start the drawing engine.
- Step 3.** Write the data in the appropriate format to the DMAWIN memory range.

After the drawing engine is started, the next successive BFIFO locations are used as the image data until the ILOAD is completed. Since the ILOAD operation generates the addresses for the destination, the addresses of the data are not used while accessing the DMAWIN window. It is recommended that host CPU instructions be used in such a way that each transfer increments the address. This way, the PCI bridge can proceed using burst transfers (assuming they are supported and enabled).

**Note:** *It is important to transfer the exact number of pixels* expected by the drawing engine, since the drawing engine will not end the ILOAD operation until all pixels have been received. A deadlock will result if the host transfers *fewer pixels* than expected to the drawing engine (the software assumes the transfer is completed, but meanwhile the drawing engine is waiting for additional data). On the other hand, if the host transfers *more pixels* than expected, the extra pixels will be interpreted by the drawing engine as register accesses.

**Note:** The ILOAD command must not be used when no data is transferred.

The total number of dwords to be transferred will differ, depending on whether or not the source is linear:

- When the source is *linear*: the data is padded at the end of the source.  

$$\text{Total} = \text{INT}(\text{psiz} * \text{width} * \text{Nlines} + 31) / 32$$
- When the source is *not linear*: the data is padded at the end of every line.  

$$\text{Total} = \text{INT}(\text{psiz} * \text{width} + 31) / 32 * \text{Nlines}$$

### **Legend:**

Total: The number of dwords to transfer  
width: The number of pixels per line to write  
Nlines: The number of lines to write  
psiz: The source size, according to [Table 5-2](#)

Table 5-2: ILOAD Source Size

bltmod	pwidth	psiz
BFCOL	PW8	8
	PW16	16
	PW24	24
	PW32	32
BMONOLEF	-	1
BMONOWF	-	1
BUYUV	-	16
BU24RGB	-	24
BU24BGR	-	24
BU32RGB	-	32
BU32BGR	-	32

### 5.5.7.1 Address Initialization

#### Linear Addresses

Register	Function	Comment / Alternate Function
<b>OPMODE</b>	Data format	A 16-bit access is required to prevent modification of the <b>dirDataSiz</b> field (bits 17:16), since direct frame buffer access may be concurrent.
<b>AR0</b>	Total number of source pixels - 1	
<b>AR3</b>	Must be 0	
<b>FXBNDRY</b>	Destination boundary (left and right)	Can use <b>FXLEFT</b> and <b>FXRIGHT</b>
<b>YDSTLEN</b>	The y start position and length	Can use <b>YDST</b> and <b>LEN</b> instead; <b>must</b> use <b>YDST</b> and <b>LEN</b> when destination address is linear (i.e.. <b>ylin</b> = 1, see <b>PITCH</b> )

#### XY Addresses

Register	Function	Comment / Alternate Function
<b>OPMODE</b>	Data format	A 16-bit access is required to prevent modification of the <b>dirDataSiz</b> field (bits 17:16).
<b>AR0</b>	Number of pixels per line - 1	
<b>AR3</b>	Must be 0	
<b>AR5</b>	Must be 0	
<b>FXBNDRY</b>	Destination boundary (left and right)	Can use <b>FXLEFT</b> and <b>FXRIGHT</b>
<b>YDSTLEN</b>	The y start position and length	Can use <b>YDST</b> and <b>LEN</b> instead; <b>must</b> use <b>YDST</b> and <b>LEN</b> when destination address is linear (i.e.. <b>ylin</b> = 1, see <b>PITCH</b> )

## 5.5.7.2 ILOAD of Two-operand Bitblt

## DWGCTL:

Res.	transc	pattern	bltmod				Res.	trans				bop				Res.	shftzero	sgnzero	arzero	solid	zmode				linear		atype		opcode			
0	0	0	+	+	+	+	0	#	#	#	#	+	+	+	+	0	1	+	0	0	0	0	0	+	+	+	+	1	0	0	1	

- **bltmod**: for a linear source, must be BFCOL. For an xy source, can be any of the following: BFCOL, BUYUV, BU32BGR, BU32RGB, BU24BGR, or BU24RGB.
- **bop**: uses any Boolean operation if **atype** is RSTR; if **atype** is RPL, **bop** must be loaded with 0000, 0011, 1100, or 1111
- **sgnzero**: can be set to '0' when **bltmod** is BFCOL, or when the **MACCESS** register's **pwidth** field is PW32; otherwise, must be '1'
- **linear**: for an xy source, must be '0'; for a linear source, must be '1'
- **atype**: can be either RPL or RSTR

	Function	Comment / Alternate Function
<b>FCOL</b>	Foreground color	For the BU32BGR and BU32RGB formats, depending on the <b>MACCESS</b> register's <b>pwidth</b> setting, the following bits from <b>FCOL</b> are used: PW32: Bits 31:24 originate from <b>forcol</b> <31:24> PW16: Bit 15 originates from <b>forcol</b> <15> when <b>dit555</b> = 1
<b>SGN</b>	Scanning direction	Must be set only when <b>sgnzero</b> = 0

There are some restrictions in the data formats that are supported for this operation. [Table 5-3](#) shows all the valid format combinations. The structure of the buffers to be transferred is defined for each data format (as shown the ‘Pixel Format’ illustrations starting on [page 5-6](#)).

**Table 5-3: ILOAD Supported Formats**

<i>Processor Type</i>	<b>bltmod</b>	<b>dmaDataSiz</b>	<b>pwidth</b>	<i>Data Format</i>
Little endian	BFCOL	00	PW8	8-bit A
			PW16	16-bit A
			PW24	24-bit A
			PW32	32-bit A
	BU24RGB	00	PW8	24-bit A
			PW16	24-bit A
			PW32	24-bit A
	BU24BGR	00	PW8	24-bit B
			PW16	24-bit B
			PW32	24-bit B
	BU32RGB	00	PW8	32-bit A
			PW16	32-bit A
PW32			32-bit A	
BU32BGR	00	PW8	32-bit B	
		PW16	32-bit B	
		PW32	32-bit B	
BUYUV	00	PW8	YUV A	
		PW16	YUV A	
	01	PW8	YUV B	
		PW16	YUV B	
Big endian	BFCOL	00	PW8	8-bit B
		01	PW16	16-bit B
		10	PW32	32-bit A
	BU32RGB	10	PW8	32-bit A
			PW16	32-bit A
			PW32	32-bit A
	BU32BGR	10	PW8	32-bit B
			PW16	32-bit B
			PW32	32-bit B
	BUYUV	00	PW8	YUV C
			PW16	YUV C
		01	PW8	YUV D
PW16			YUV D	
		PW32	YUV D	



### 5.5.7.3 ILOAD with Expansion (Character Drawing)

#### DWGCTL:

Res.	transc	pattern	bltmod				Res.	trans				bop				Res.	shftzero	sgnzero	arzero	solid	zmode				linear	atype			opcode			
0	+	0	+	+	+	+	0	+	+	+	+	+	+	+	+	0	1	1	0	0	0	0	0	1	+	+	+	1	0	0	1	

- **bltmod**: must be set to either BMONOLEF or BMONOWF
- **trans**: if **atype** is BLOCK, the transparency pattern is not supported - the value of **trans** must be '0000'
- **bop**: uses any Boolean operation if **atype** is RSTR; if **atype** is RPL, **bop** must be loaded with 0000, 0011, 1100, or 1111; if **atype** is BLOCK, **bop** must be loaded with 1100
- **atype**: must be set to either RPL, RSTR, or BLOCK

Register	Function	Comment / Alternate Function
BCOL	Background color	Only when <b>transc</b> = '0'
FCOL	Foreground color	

 Note that the **MACCESS** register's **pwidth** field can be set to 24 bits per pixel (PW24) with the following limitations:

- **atype** is either RPL or RSTR
- or*
- **forcol**<31:24>, **forcol**<23:16>, **forcol**<15:8>, and **forcol**<7:0> are set to the same value, and **backcol**<31:24>, **backcol**<23:16>, **backcol**<15:8>, and **backcol**<7:0> are set to the same value.

There are some restrictions in the data formats that are supported for this operation. Table 5-4 shows all the valid format combinations. The structure of the buffers to be transferred is defined for each data format (as shown the 'Pixel Format' illustrations starting on page 5-6).

*Table 5-4: Bitblt with Expansion Supported Formats*

Processor Type	bltmod	dmaDataSiz	Data Format
Little endian	BMONOLEF	00	MONO A
	BMONOWF	00	MONO B
Big endian	BMONOWF	00	MONO C

### 5.5.8 Scaling

The MGA-2064W supports scaling operations, which are divided into two phases: horizontal scaling and vertical scaling. Each phase is executed in a different manner.

- Horizontal scaling uses ILOAD\_SCALE (performs pixel replication) or ILOAD\_FILTER (performs minimum filtering when scaling). The following operations are supported for horizontal scaling:
  - Up scaling (down scaling is not supported). The minimum scaling factor is 2x when ILOAD\_FILTER is used.
  - Pixel re-formatting. There are some restrictions in the data formats that are supported for this operation. Table 5-5 shows all the valid format combinations. The structure of the buffers to be transferred is defined for each data format (as shown the ‘Pixel Format’ illustrations starting on page 5-6).

*Table 5-5: Scaling Supported Formats*

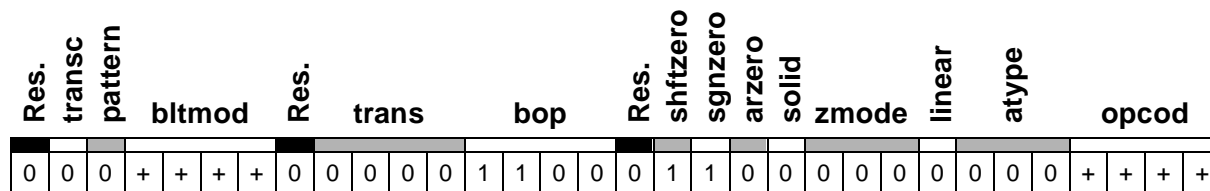
<i>Processor Type</i>	<b>bltmod</b>	<b>dmaDataSiz</b>	<b>pwidth</b>	<i>Data Format</i>
Little endian	BU24RGB	00	PW8 - PW16 - PW32	24-bit A
	BU24BGR	00	PW8 - PW16 - PW32	24-bit B
	BU32RGB	00	PW8 - PW16 - PW32	32-bit A
	BU32BGR	00	PW8 - PW16 - PW32	32-bit B
	BUYUV	00	PW8 - PW16 - PW32	YUV A
	BUYUV	01	PW8 - PW16 - PW32	YUV B
Big endian	BU32RGB	10	PW8 - PW16 - PW32	32-bit A
	BU32BGR	10	PW8 - PW16 - PW32	32-bit B
	BUYUV	00	PW8 - PW16 - PW32	YUV C
	BUYUV	01	PW8 - PW16 - PW32	YUV D

- Vertical scaling is performed using the FBITBLT or BITBLT functions. FBITBLT should be used except when certain restrictions are not met (see Section 5.5.6.1).

The following steps must be executed for scaling:

- Step 1.** Initialize the scaling engine as specified in subsection 5.5.8.1. Also, remember to program the registers listed in section 5.5.2. *Do not start the drawing engine.*
- Step 2.** Initialize the drawing engine for horizontal scaling. The last register you program must be accessed in the 1D00-1DFFh range in order to start the drawing engine.

**DWGCTL:**



- **bltmod**: can be set to BUYUV, BU32BGR, BU32RGB, BU24BGR, or BU24RGB
- **opcode**: can be set to ILOAD\_SCALE or ILOAD\_FILTER

<i>Register / Space</i>	<i>Field</i>	<i>Comment / Alternate Function</i>
<b>LEN</b>	Number of lines	Without line replication

- Step 3.** Send the data that is to be used in the scaling process. [Table 5-5](#) shows the various supported data formats. As with normal ILOAD operations (see the Note on [page 5-35](#)), the exact amount of data must be transferred. The amount of data is derived from the following formula (data must be padded on every line):

$$\text{Total} = \text{INT}((\text{psiz} * \text{width} + 31) / 32) * \text{Nlines}$$

**Legend:**

- Total: The number of dwords to transfer  
width: The number of pixels per line to write  
Nlines: The number of lines to write  
psiz: The source size, according to [Table 5-6](#)

**Table 5-6: Scaling Source Size**

<b>bltmod</b>	<b>psiz</b>
BUYUV	16
BU24RGB	24
BU24BGR	24
BU32RGB	32
BU32BGR	32

- Step 4.** Initialize the drawing engine for vertical scaling. The last register you program must be accessed in the 1D00-1DFFh range in order to start the drawing engine.

<i>Register / Space</i>	<i>Function</i>	<i>Comment / Alternate Function</i>
<b>LEN</b>	Number of lines	Replicated lines
<b>DWGCTL</b>	040A600Ch (FBITBLT) or 040C6008h (BITBLT)	

- Step 5.** Repeat Steps 2 to 4 until the end of the scaling sequence.

### 5.5.8.1 Scaling Initialization

#### ILOAD\_SCALE

<i>Register</i>	<i>Function</i>	<i>Comment / Alternate Function</i>
<b>OPMODE</b>	Data format	A 16-bit access is required to prevent modification of the <b>dirDataSiz</b> field (bits 17:16).
<b>AR0</b>	DST_END_ADDRESS - DST_Y_INCREMENT	
<b>AR2</b>	SOURCE_X_DIMENSION	
<b>AR3</b>	DST_START_ADDRESS - DST_Y_INCREMENT	
<b>AR5</b>	DST_Y_INCREMENT	Only required if vertical scaling is used
<b>AR6</b>	SOURCE_X_DIMENSION - DESTINATION_X_DIMENSION	
<b>FXBNDRY</b>	Destination boundary (left and right)	Can use <b>FXLEFT</b> and <b>FXRIGHT</b>
<b>YDST</b>	y start position	

#### ILOAD\_FILTER

<i>Register</i>	<i>Function</i>	<i>Comment / Alternate Function</i>
<b>OPMODE</b>	Data format	A 16-bit access is required to prevent modification of the <b>dirDataSiz</b> field (bits 17:16).
<b>AR0</b>	DST_END_ADDRESS - DST_Y_INCREMENT	
<b>AR2</b>	$(2 * \text{SOURCE\_X\_DIMENSION} - 1)$	
<b>AR3</b>	DST_START_ADDRESS - DST_Y_INCREMENT	
<b>AR5</b>	DST_Y_INCREMENT	Only required if vertical scaling is used
<b>AR6</b>	$(2 * \text{SOURCE\_X\_DIMENSION} - 1) -$ DESTINATION_X_DIMENSION	
<b>FXBNDRY</b>	Destination boundary (left and right)	Can use <b>FXLEFT</b> and <b>FXRIGHT</b>
<b>YDST</b>	y start position	

## 5.5.9 IDUMP Programming

The following subsections list the registers that must be specifically programmed for IDUMP (image dump: VRAM -> Host) operations. You must take the following steps:


**Step 1.** Initialize the registers. Remember to program the registers listed in section 5.5.2.

### DWGCTL:

Res.	transc	pattern	bltmod	Res.	trans	bop	Res.	shftzero	sgnzero	arzero	solid	zmode	linear	atype	opcode									
0	0	0	+	+	+	+	0	0	0	0	0	1	1	0	0	0	1	1	0	0	1	0	1	0

■ **bltmod**: can be BU32BGR, BU32RGB, BU24BGR, or BU24RGB. See Table 5-8.

Register	Function	Comment / Alternate Function
<b>OPMODE</b>	Data format	A 16-bit access is required to prevent modification of the <b>dirDataSiz</b> field (bits 17:16). There is no need to program the <b>dmamod</b> field of the <b>OPMODE</b> register - reading the DMAWIN is sufficient to trigger the IDUMP.
<b>AR0</b>	Source end address	
<b>AR3</b>	Source start address	
<b>AR5</b>	Source y increment	Not required for a linear source
<b>FXBNDRY</b>	Destination boundary. Left = 0; Right = number of pixels per line minus 1	Can use <b>FXLEFT</b> and <b>FXRIGHT</b>
<b>YDSTLEN</b>	The y start position and number of lines	

 **PITCH**: The **ylin** field of this global initialization register must be set to '0'. The pitch value itself is not used.

**Step 2.** Program the last register to access the 1D00-1DFFh range in order to start the drawing engine.

**Step 3.** Read the data in the appropriate format from the DMAWIN memory range.

Since the IDUMP operation generates the addresses for the destination, the addresses of the data are not used while accessing the DMAWIN window. Subsequently, move string instructions can be used through the 7K space of DMAWIN to read the data from the MGA-2064W. It is recommended that host CPU instructions be used in such a way that each transfer increments the address. This way, the PCI bridge can proceed using burst transfers (assuming they are supported and enabled).

Dwords are always transferred in whole numbers: depending on the source's width and alignment, part of the last dword of every line transferred may contain irrelevant data. The total number of dwords can be calculated by the following formula:

$$\text{Total} = \text{INT} ((\text{psiz} * \text{width} + 31) / 32) * \text{Nlines}$$

**Legend:**

- Total: The number of dwords to transfer  
width: The number of pixels to be read in the x direction  
Nlines: The number of lines to read  
psiz: The destination size, according to [Table 5-7](#)

**Table 5-7: IDUMP Source Size**

<b>bltmod</b>	<b>pwidth</b>	<b>psiz</b>
BU32RGB	PW8	8
	PW16	16
	PW24	24
	PW32	32
BU32BGR	-	32
BU24RGB	-	24
BU24BGR	-	24

There are some restrictions in the data formats that are supported for this operation. [Table 5-8](#) shows all the valid format combinations. The structure of the buffers to be transferred is defined for each data format (as shown the 'Pixel Format' illustrations starting on [page 5-6](#)).

**Table 5-8: IDUMP Supported Formats**

<b>Processor Type</b>	<b>bltmod</b>	<b>dmaDataSiz</b>	<b>pwidth</b>	<b>Data Format</b>
Little endian	BU32RGB	00	PW8	8-bit A
			PW16	16-bit A
			PW24	24-bit A
			PW32	32-bit A
	BU32BGR	00	PW32	32-bit B
BU24RGB	00	PW32	24-bit A	
BU24BGR	00	PW32	24-bit B	
Big endian	BU32RGB	00	PW8	8-bit B
		01	PW16	16-bit B
		10	PW32	32-bit A
	BU32BGR	10	PW32	32-bit B

## 5.6 CRTC Programming

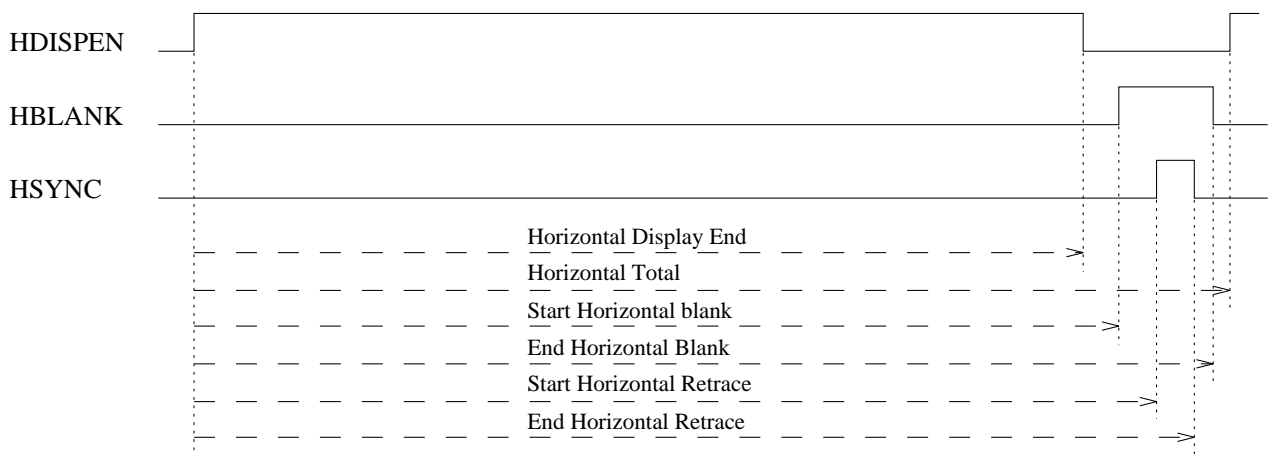
The CRTC can be programmed in one of two modes: VGA mode or Power Graphic mode. The **mgamode** field of the **CRTCEXT3** register is used to select the operating mode.

CRTC registers 0 to 7 can be write-protected by the **crtcprotect** field of the **CRTC11** register.

In VGA mode, all of the **CRTC** extension bits must be at '0'. The **page** field of **CRTCEXT4** can be used to select a different page of RAM in which to write pixels. As well, the **interleave** field of the **OPTION** register must be set to '0'.

### 5.6.1 Horizontal Timing

*Figure 5-2: CRTC Horizontal Timing*



In VGA mode, the horizontal timings are defined by the following VGA registers:

- htotal<7:0>** Horizontal total. Should be programmed with the total number of displayed characters plus the non-displayed characters minus 5.
- hdispnd<7:0>** Horizontal display end. Should be loaded with the number of displayed characters - 1.
- hblkstr<7:0>** Start horizontal blanking
- hblkend<6:0>** End horizontal blanking. Should be loaded with (**hblkstr** + Horizontal Blank signal width) AND 3Fh. Bit 6 is not used in VGA mode (**mgamode** = 0)
- hsyncstr<7:0>** Start horizontal retrace
- hsyncend<4:0>** End horizontal retrace. Should be loaded with (**hsyncstr** + Horizontal Sync signal width) AND 1Fh.
- hsyncdel<1:0>** Horizontal retrace delay.

In Power Graphic mode, the following bits are extended to support a wider display area:

- htotal<8:0>** Horizontal total
- hblkstr<8:0>** Start horizontal blanking
- hsyncstr<8:0>** Start horizontal retrace

The horizontal counter can be reset in Power Graphic mode by a rising edge on the **VIDRST** pin, if the **hrsten** bit of the **CRTCEXT1** register is set to '1'.

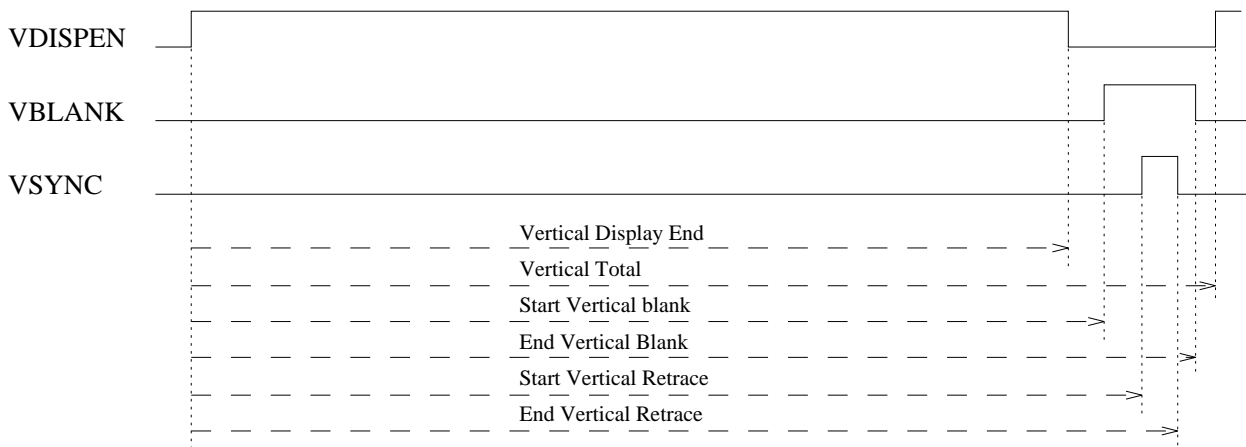
The units of the horizontal counter are ‘character clocks’ for VGA mode, or 8 pixels in Power Graphic mode. The **scale** field of the **CRTCEXT3** register is used to bring the VCLK clock down to an ‘8 pixel’ clock.

The suggested scale factor is shown in the following table:

interleave	Bits/Pixel	Scale
0	8	001
	16	011
	24	101
	32	111
1	8	000
	16	001
	24	010
	32	011

## 5.6.2 Vertical Timing

*Figure 5-3: CRTC Vertical Timing*



In VGA mode, the vertical timings are defined by the following VGA registers:

- vtotal<9:0>** Vertical total. Should be programmed with the total number of displayed lines plus the non-displayed lines minus 2.
- vdispnd<9:0>** Vertical display end. Should be loaded with the number of displayed lines minus 1.
- vblkstr<9:0>** Start vertical blanking. The programmed value is one less than the horizontal scan line count at which the vertical blanking signal becomes active.
- vblkend<7:0>** End vertical blanking. Should be loaded with (**vblkstr** - 1 + Vertical Blank signal width) AND FFh.
- vsyncstr<9:0>** Start vertical retrace
- vsyncend<3:0>** End vertical retrace. Should be loaded with (**vsyncstr** + Vertical Sync signal width) AND 0Fh.
- linecomp<9:0>** Line compare



In Power Graphic mode, the following bits are extended to support a larger display area:

<b>vtotal&lt;11:0&gt;</b>	Vertical total
<b>vdispnd&lt;10:0&gt;</b>	Vertical display end
<b>vblkstr&lt;11:0&gt;</b>	Vertical blanking start
<b>vsyncstr&lt;11:0&gt;</b>	Start vertical retrace
<b>linecomp&lt;10:0&gt;</b>	Line compare

The units of the vertical counter can be 1 or 2 scan lines, depending on the value of the **hsyncsel** bit of the **CRTC17** register.

The Vertical counter can be reset in Power Graphic mode by the **VIDRST** pin if the **vrsten** bit of the **CRTCEXT1** register is set to '1'. The **vinten** and **vintclr** fields of the **CRTC11** register can be used to control the vertical interrupt.

### 5.6.3 Memory Address Counter

In VGA mode, the following registers are used to program the memory address counter and the cursor/underline circuitry:

<b>stradd&lt;15:0&gt;</b>	Start address
<b>offset&lt;7:0&gt;</b>	Logical line width of the screen. This is programmed with the number of double or single words in one character line.
<b>curpos&lt;15:0&gt;</b>	Cursor position
<b>prowsan&lt;4:0&gt;</b>	Preset row scan
<b>maxscan&lt;4:0&gt;</b>	Maximum scan line
<b>currowstr&lt;4:0&gt;</b>	Row scan cursor begins
<b>currowend&lt;4:0&gt;</b>	Row scan cursor ends
<b>curoff&lt;4:0&gt;</b>	Cursor off
<b>undrow&lt;4:0&gt;</b>	Horizontal row scan where underline will occur
<b>curskew&lt;1:0&gt;</b>	Cursor skew control

- The row scan counter can be clocked by the horizontal sync signal or by the horizontal sync signal divided by 2, depending on the value of the **conv2t4** (200 to 400 line conversion) field of the **CRTC9** register.
- The memory address counter clock is controlled by **count4 (CRTC14)** and **count2 (CRTC17)**. These fields have no effect in Power Graphic mode.
- The memory address can be modified by the **dword (CRTC14)**, **wbmode**, **addwrap**, **selrowscan**, and **cms (CRTC17)** fields.

In Power Graphic mode, the following bits are extended in order to support a larger display, and up to 8 Mbytes of memory.

- stradd<19:0>** Start address.
- offset<9:0>** Logical line width of the screen. This is programmed with the number of slices in one character line.
- The display can be placed in interlace mode if the **interlace** bit of the **CRTCEXT0** register is set to '1'.
  - The **curpos**, **prowsan**, **currowstr**, **currowend**, **curoff**, **undrow** and **curskew** registers are not used in Power Graphic mode.
  - The **maxscan** field of the **CRTC9** register is used to zoom vertically in Power Graphic mode.
  - Horizontal zooming can be achieved by dividing the pixel clock period and reprogramming the horizontal registers.

#### 5.6.4 Programming in VGA Mode

The VGA CRTC of the MGA-2064W chip conforms to VGA standards. The limitations listed below need only be taken into account when programming extended VGA modes.

##### Limitations:

- **htotal** must be greater than 0.
- **vtotal** must be greater than 0.
- **htotal** - **hdispen** must be greater than 0
- In interlace mode, **htotal** must be equal to or greater than **hsyncend** + 1
- **htotal** - **bytepan** + 2 must be greater than **hdispend**
- **hsyncstr** must be greater than **hdispend** + 2

##### CRTC Latency Formulas

This section presents several rules that must be followed in VGA mode in order to adhere to the latency constraints of the MGA-2064W's CRTC.

In the formulas below, 'cc' represents the number of VCLKs per character. The display modes are controlled by the **SEQ1** register's **dotmode** and **dotclkrt** fields and the **ATTR10** register's **pelwidth** field as shown below:

<i>Display Mode</i>	<b>dotmode</b>	<b>dotclkrt</b>	<b>pelwidth</b>	<i>cc</i>
Character mode: 8	1	0	0	8
Character mode: 9	0	0	0	9
Zoomed character: 16	1	1	0	16
Zoomed character: 18	0	1	0	18
Graphics (non-8 bit/pixel)	1	0	0	8
Zoomed graphics (non-8 bit/pixel)	1	1	0	16
Graphics (8 bit/pixel)	1	0	1	4
Zoomed graphics (8 bit/pixel)	1	1	1	8

The following factors (in GCLKs) must be applied to the formulas below, according to whether text or graphics are being displayed:

<i>Variable</i>	<i>VGA Text</i>	<i>VGA Graphics</i>
A	27	19
B	7	1
C	9	8
D	57	30

Using these values, we can determine the following rules:

1.  $(cc * ((H_{total} - Byte_{pan}) - (H_{dispend} + \text{MAX}(H_{dispskew} + 2, H_{syncstr} - H_{dispend})) + 1) - 3) * Tvclk \geq A * Tgclk$
2.  $(cc * 4 - 1) * Tvclk \geq A * Tgclk$
3.  $cc * Tvclk \geq B * Tgclk$
4.  $(cc * ((H_{total} - Byte_{pan}) - H_{dispend} + 2) - 1) * Tvclk \geq (A + C) * Tgclk$
5.  $(cc * ((H_{total} - Byte_{pan}) - (H_{dispend} + \text{MAX}(H_{dispskew} + 2, H_{syncstr} - H_{dispend})) + 2) - 3) * Tvclk \geq (A + C) * Tgclk$
6.  $(cc * ((H_{total} - Byte_{pan}) - H_{dispend} + 3) - 1) * Tvclk \geq (D + C) * Tgclk$

### 5.6.5 Programming in Power Graphic Mode

The horizontal and vertical registers are programmed as for VGA mode, and they can use the **CRTC** extension fields.

The memory address mapper must be set to byte mode and the offset register value (**CRTC13**) must be programmed with the following formulas:

$$\text{offset} = \frac{\text{pitch}}{\text{Fbpp} * \text{Fileave}}$$

Where: pitch is the line pitch in pixels, and

<i>bpp</i>	<i>Fbpp</i>	<i>interleave</i>	<i>Fileave</i>
8	8	0	1
16	4	1	2
24	2 - 2/3		
32	2		

For example, in an interleave system with a 16 bit/pixel frame buffer at a resolution of 1280 x 1024:

$$\text{offset} = 1280 / (2 * 4) = 160$$

The value that is programmed into **offset** must be a multiple of 8. This means, in other words, that the pitch of the screen must be a multiple of the following:

<i>bpp</i>	<i>Non-interleave</i>	<i>Interleave</i>
8	128	64
16	64	32
24	42 2/3	21 1/3
32	32	16

The **startadd** field represents the number of pixels to offset the start of the display by.

<i>bpp</i>	<i>Non-interleave</i>	<i>Interleave</i>
8	4	8
16	2	4
24	1 1/3	2 2/3
32	1	2

For example, in a non-interleave system with a 16 bit/pixel frame buffer, **startadd** = 16 indicates that the display will be offset by 32 pixels.

You should remember that the memory address counter is used to generate data transfer requests to the MCTL in Power Graphic mode. It is *not* used to get data from the RAM to the Attributes Controller (**ATTR**) as in VGA mode.

There is no overscan in Power Graphic mode because the blank/ and cde pins on the RAMDAC are tied together with the blank/ signal from the MGA-2064W chip. So:

$$\begin{aligned} \mathbf{htotal+5} &== \mathbf{hblkend+1} \\ \mathbf{hdispnd+1} &== \mathbf{hblkstr+1} \end{aligned}$$

The End Horizontal Blank value must always be greater than **hsyncstr** + 1, so that the start address latch can be loaded before the memory address counter.

A composite sync (block sync) can be generated on the HSYNC pin of the chip if the **csyncen** field of the **CRTCEXT3** register is set to '1'. The VSYNC pin will continue to carry the vertical retrace signal.

The composite sync is always active low. Note that the following values must be programmed in Power Graphic mode.

- **hsyncsel** = 0
- **hdispskew** = 0
- **hsyncsel** = 0
- **bytepan** = 0
- **conv2t4** = 0
- **dotclkrt** = 0
- **dword** = 0, **wbmode** = 1 (refer to the 'Byte access' table in the **CRTC17** register description)
- **selrowscan** = 1, **cms** = 1

### Changing the Start Address

Care must be taken when changing the start address. To avoid artifacting when changing the start address, the following conditions should be respected:

- A change to the start address will only take effect at the beginning of the next frame. In a page flipping application, when the start address is modified, it is important that any modification to the currently-displayed frame be prevented until the vertical blank period.
- The start address register is broken into three parts. In order to ensure that the address is consistent, these locations should not be updated in the line which precedes the first display line of the screen.
- If any of the four LSBs of the start address must be modified, then the start address should be modified during the vertical blanking interval.

In all cases, you can see which line the CRTC is currently displaying (including blank lines) by reading the **VCOUNT** register.

## Interlace Mode

If interlace is selected, the offset value must be multiplied by 2.

- The **vtotal** value must be the total number of lines (of both fields) divided by 2.  
For example, for a 525 line display, **vtotal** = 260.
- The **vsyncstr** value must be divided by 2
- The **vblkstr** values must be divided by 2
- The **hvidmid** field must be programmed to become active exactly in the middle of a horizontal line.

## Zooming

Horizontal zooming is achieved by slowing down the pixel clock and re-programming the horizontal registers of the CRTC.

- For example, to obtain a horizontal zoom rate of x2, slow the pixel clock by two and re-program all of the horizontal CRTC registers so that the period, active time, front and back porch, blank and sync width (in ns) remain the same. The horizontal counter will have a precision of 16 or 32 pixels on the screen for zoom rates of x2 or x4 respectively.

Vertical zooming is achieved by re-scanning a line 'n' times. Program the **CRTC9** register's **maxscan** field with the appropriate value, n-1, to obtain a vertical zoom.

- For example, set **maxscan** = 3 to obtain a vertical zoom rate of x4.

## Limitations:

- **htotal** must be greater than 0 (because of the delay registers on the **htotal** comparator)
- **htotal** - **hdispen** must be greater than 0
- In interlace mode, **htotal** must be equal to or greater than **hsyncend** + 1.
- **htotal** - **bytepan** + 2 must be greater than **hdispend**
- **hsyncstr** must be greater than **hdispend** + 2
- **vtotal** must be greater than 0 (because of the delay registers on the **vtotal** comparator)
- In interlace mode, **vtotal** must be an even number.

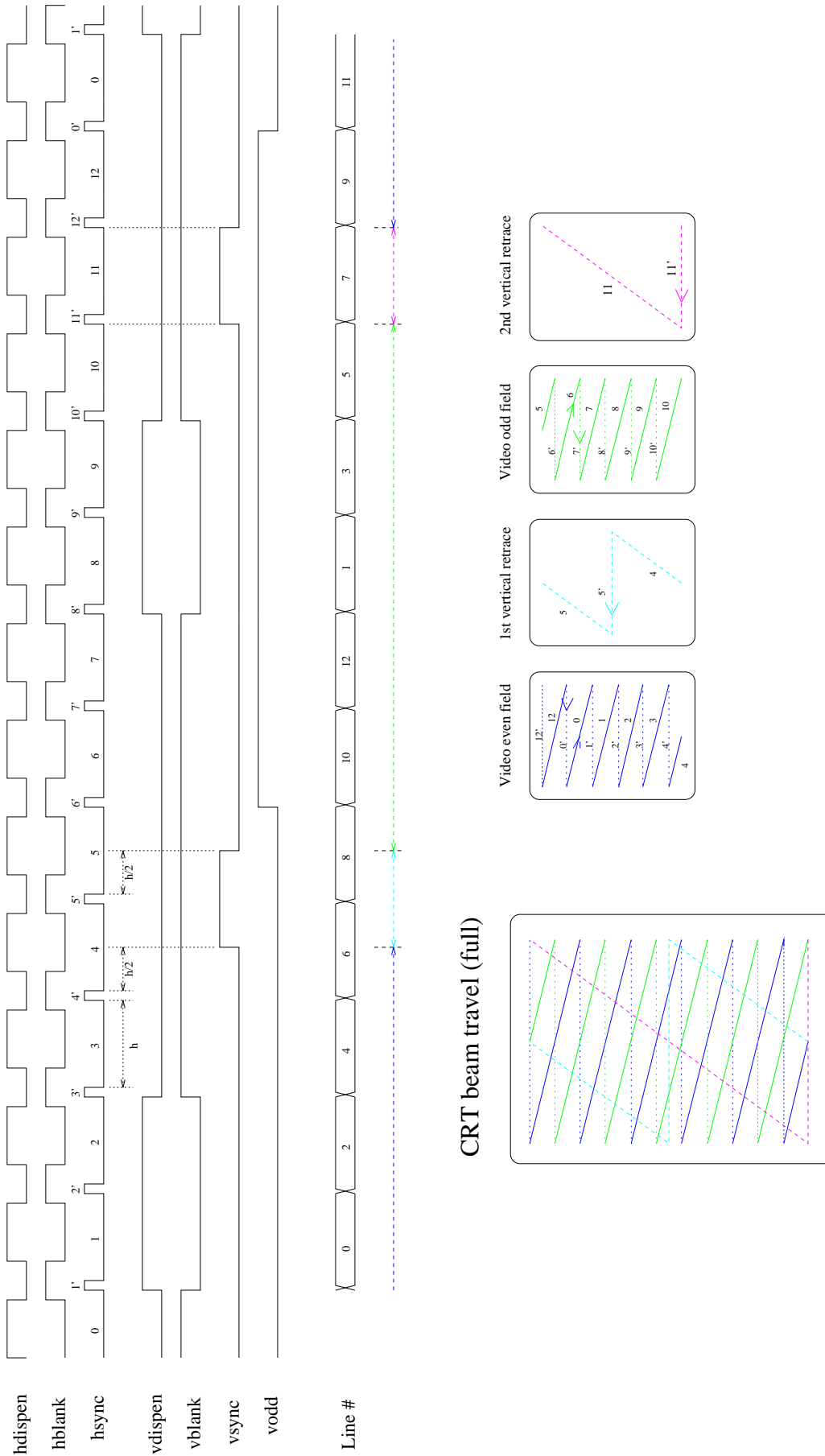
## CRTC Latency Formulas

This section presents several rules that must be followed in Power graphic mode in order to adhere to the latency constraints of the MGA-2064W's CRTC.

In the formulas below, 'cc' represents the number of VCLKs per character (8 pixels). Using these values, we can determine the following rules:

1.  $(cc * (H\_total - (H\_dispend + \text{MAX}(\text{startadd}<3:0> + 1/cc, H\_syncstr - H\_dispend))) - 1.5) * Tvclk \geq 51 * Tgclk$
2.  $58.5 * Tvclk \geq 51 * Tgclk$
3.  $16 * Tvclk \geq Tgclk$
4.  $(cc * (H\_total - H\_dispend) + \text{MOD}(\text{pitch}*cc/8 - 1, 16) + 1.5) * Tvclk \geq 62 * Tgclk$
5.  $(cc * (H\_total - (H\_dispend + \text{MAX}(\text{startadd}<3:0> + 1/cc, H\_syncstr - H\_dispend)) + 1) - 1.5) * Tvclk \geq 59 * Tgclk$
6.  $(cc * (H\_total - H\_dispend + 1) + \text{MOD}(\text{pitch}*cc/8 - 1, 16) + 1.5) * Tvclk \geq 70 * Tgclk$

Figure 5-4: Video Timing in Interlace Mode



## 5.7 Interrupt Programming

The MGA-2064W has four interrupt sources:

### 1. Pick interrupt

This interrupt is used to help with item selection in a drawing. A rectangular pick region is programmed using the clipper registers (**YTOP**, **YBOT**, **CXLEFT**, **CXRIGHT**). All planes must be masked by writing FFFFFFFFh to the **PLNWT** register. The drawing engine then redraws every primitive in the drawing. When pixels are output in the clipped region, the pick pending status is set. After a primitive has been initialized, the **STATUS** register's **dwgengsts** bit can be polled to determine if some portion of the primitive lies within the clipping region.

Picking interrupts are generated when primitives are drawn using either RPL, RSTR, ZI, or I. These access types are explained in the **atype** field description for the **DWGCTL** register in [Chapter 4](#).

### 2. Vertical sync interrupt

This interrupt is generated every time the vsync signal goes active. It can be used to synchronize a process with the video raster such as frame by frame animation, etc. The vsync interrupt enable and clear are both located in the **CRTC11** VGA register.

### 3. Vertical line interrupt

This interrupt is generated when the value of the **linecomp** field of **CRTC18** equals the current vertical count value. This interrupt is more flexible than the vertical sync interrupt because it allows interruption on any horizontal line (including blank and sync lines).

### 4. External interrupt

This interrupt is generated when the external interrupt line is driven active. It is the responsibility of the external device to provide the clear and enable enable functions.

The following table summarizes the supported functionality that is associated with each interrupt source.

<i>Interrupt</i>	<i>STATUS</i>	<i>EVENT</i>	<i>ENABLE</i>	<i>CLEAR</i>
Pick	- -	<b>pickpen</b> <b>STATUS&lt;2&gt;</b>	<b>pickien</b> <b>IEN&lt;2&gt;</b>	<b>pickiclr</b> <b>ICLEAR&lt;2&gt;</b>
Vertical sync	<b>vsyncsts</b> <b>STATUS&lt;3&gt;</b>	<b>vsyncpen</b> <b>STATUS&lt;4&gt;</b>	<b>vinten</b> <b>CRTC11&lt;5&gt;</b>	<b>vintclr</b> <b>CRTC11&lt;4&gt;</b>
Vertical line	- -	<b>vlinepen</b> <b>STATUS&lt;5&gt;</b>	<b>vlineien</b> <b>IEN&lt;5&gt;</b>	<b>vlineiclr</b> <b>ICLEAR&lt;5&gt;</b>
External	<b>extpen</b> <b>STATUS&lt;6&gt;</b>	-	<b>extien</b> <b>IEN&lt;6&gt;</b>	-

**STATUS** Indicates which bit reports the current state of the interrupt source.

**EVENT** Indicates which bit reports that the interrupt event has occurred.

**ICLEAR** A pending bit is kept set until it is cleared by the associated clear bit.

**IEN** An interrupt source may or may not take part in activating the **PINTA**/ hardware interrupt line. The **EVENT** and **STATUS** flags are not affected by interrupt enabling or disabling.

**Notes:**

- It is a good practice to clear an interrupt before enabling it
- **vsyncpen** is set on the rising edge of vsync
- **vsyncpen** is set on the first pixel within the clipping box
- **vlinepen** is set at the beginning of the line



## Alphabetical Listing

### Power Graphic Mode Fields (includes configuration space and memory space register fields)

ar0<17:0>	4-17	dr6<23:0>	4-32
ar1<23:0>	4-18	dr7<23:0>	4-33
ar2<17:0>	4-19	dr8<23:0>	4-34
ar3<23:0>	4-20	dwgengsts<16>	4-65
ar4<17:0>	4-21	eepromwt<20>	4-14
ar5<17:0>	4-22	extien<6>	4-53
ar6<17:0>	4-23	extpen<6>	4-65
arzero<12>	4-42	fastbackcap RO <23>	4-4
atype<6:4>	4-41	fifocount<5:0>	4-48
backcol<31:0>	4-24	forcol<31:0>	4-47
bempty<9>	4-48	funcnt<6:0>	4-63
bfull<8>	4-48	funoff<21:16>	4-63
biosenR/W<30>	4-14	fxleft<15:0>	4-49
bltmod<28:25>	4-45	fxleft<15:0>	4-50
bop<19:16>	4-43	fxright<15:0>	4-51
busmaster RO <2>	4-4	fxright<31:16>	4-49
class<31:8>	4-3	header<23:16>	4-7
cxleft<10:0>	4-25	index<13:2>	4-10
cxleft<10:0>	4-26	interleave<12>	4-13
cxright<10:0>	4-27	intline R/W <7:0>	4-8
cxright<26:16>	4-25	intpin RO <15:8>	4-8
cybot<22:0>	4-70	iospace R/W <0>	4-4
cytop<22:0>	4-74	iy<11:0>	4-59
data<31:0>	4-9	length<15:0>	4-54
detparerrRO <31>	4-5	length<15:0>	4-72
device<31:16>	4-6	linear<7>	4-41
devseltimRO <26:25>	4-5	memreset<15>	4-55
dirDataSiz<17:16>	4-56	memspace R/W <1>	4-4
dit555<31>	4-55	memspace-indRO <0>	4-11
dmaDataSiz<9:8>	4-56	memspace-indRO <0>	4-12
dmamod<3:2>	4-56	mgabase1<31:14>	4-11
dr0<31:0>	4-28	mgabase2<31:23>	4-12
dr10<23:0>	4-35	nodither<30>	4-55
dr11<23:0>	4-36	nogscale<21>	4-14
dr12<23:0>	4-37	noretry<29>	4-14
dr14<23:0>	4-38	opcod<3:0>	4-40
dr15<23:0>	4-39	patreg<63:0>	4-58
dr2<31:0>	4-29	pattern<29>	4-46
dr3<31:0>	4-30	pickiclr<2>	4-52
dr4<23:0>	4-31	pickien<2>	4-53
		pickpen<2>	4-65
		plnwrmsk<31:0>	4-60
		powerpc<31>	4-14
		prefetchableRO <3>	4-11
		prefetchableRO <3>	4-12

## Alphabetical Listing

### (Power Graphic Mode Fields, con't)

productidRO <28:24>	4-14	y_off<6:4>	4-63
pwidth<1:0>	4-55	y_start<31:16>	4-69
resparerr RO <6>	4-4	ydst<21:0>	4-71
revision<7:0>	4-3	ydstorg<22:0>	4-73
rfhcnt<19:16>	4-13	ylin<15>	4-59
rombase<31:16>	4-15	yval<31:16>	4-72
romen<0>	4-15	zmode<10:8>	4-41
scanleft<0>	4-62	zorg<22:0>	4-75
sdxl<1>	4-62		
sdxr<5>	4-62		
sdyl<2>	4-62		
sdycl<0>	4-62		
sellin<31:29>	4-71		
SERRenable RO <8>	4-4		
sgnzero<13>	4-42		
shftzero<14>	4-43		
sigsyserrRO <30>	4-5		
sigtargab RO <27>	4-5		
softreset<0>	4-61		
solid<11>	4-42		
spage <25:24>	4-20		
specialcycle RO <3>	4-4		
srcreg<127:0>	4-64		
stylelen<22:16>	4-63		
trans<23:20>	4-44		
transc<30>	4-46		
type RO<2:1>	4-11		
typeRO <2:1>	4-12		
vcount<11:0>	4-66		
vendor<15:0>	4-6		
vgaioen<8>	4-13		
vgasnoop R/W <5>	4-4		
vlineiclr<5>	4-52		
vlineien<5>	4-53		
vlinepen<5>	4-65		
vsyncpen<4>	4-65		
vsyncsts<3>	4-65		
waitcycle RO <7>	4-4		
x_end<15:0>	4-68		
x_off<3:0>	4-63		
x_start<15:0>	4-69		
xdst<15:0>	4-67		
y_end<31:16>	4-68		

# Alphabetical Listing

## VGA Mode Fields

addwrap<5>	4-116	funsel<4:3>	4-135
asynrst<0>	4-147	gcgrmode<0>	4-139
atcgrmode<0>	4-83	gcoddevmd<4>	4-137
attradssel<7>	4-119	gctld<15:8>	4-131
attrd<15:8>	4-81	gctlx<3:0>	4-131
attrx<4:0>	4-120	hblkend<4:0>	4-94
attrx<4:0>	4-80	hblkend<6>	4-123
blinken<3>	4-83	hblkend<7>	4-96
bytepan<6:5>	4-99	hblkstr<1>	4-123
chain4<3>	4-151	hblkstr<7:0>	4-93
chainoddeven<1>	4-139	hdispend<7:0>	4-92
clkssel<3:2>	4-144	hdispskew<6:5>	4-94
cms<0>	4-114	hpelcnt<3:0>	4-87
colcompen<3:0>	4-140	hpgoddev<5>	4-144
colplen<3:0>	4-86	hretrace<0>	4-143
colsel54<1:0>	4-88	hrsten<3>	4-123
colsel76<3:2>	4-88	hsyncdel<6:5>	4-96
conv2t4<7>	4-100	hsyncend<4:0>	4-96
count2<3>	4-116	hsyncoff<4>	4-123
count4<5>	4-111	hsyncpol<6>	4-145
cpudata<7:0>	4-118	hsyncsel<2>	4-116
crtcd<15:8>	4-90	hsyncstr<2>	4-123
crtcxtd<15:8>	4-121	hsyncstr<7:0>	4-95
crtcctx<2:0>	4-121	htotal<0>	4-123
crtcintCRT<7>	4-142	htotal<7:0>	4-91
crtcprotect<7>	4-108	hvidmid<7:0>	4-128
crtcstN<7>	4-116	interlace<7>	4-122
crtcx<5:0>	4-89	ioaddsel<0>	4-144
csyncen<6>	4-125	lgren<2>	4-83
curloc<7:0>	4-105	linecomp<4>	4-98
curloc<7:0>	4-106	linecomp<6>	4-100
curoff<5>	4-101	linecomp<7:0>	4-117
currowend<4:0>	4-102	linecomp<7>	4-124
currowstr<4:0>	4-101	mapasel<5,3:2>	4-150
curskew<6:5>	4-102	mapbsel<4,1:0>	4-150
diag<5:4>	4-143	maxscan<4:0>	4-100
dotclkrt<3>	4-148	memmapsl<3:2>	4-139
dotmode<0>	4-148	memsz256<1>	4-151
dsts<1:0>	4-129	mgamode<7>	4-126
dword<6>	4-111	mode256<6>	4-138
featcb0<0>	4-130	mono<1>	4-83
featcb1<1>	4-130	offset<5:4>	4-122
featin10<6:5>	4-142	offset<7:0>	4-110
		ovscol<7:0>	4-85
		p5p4<7>	4-84

# Alphabetical Listing

---

## VGA Mode Fields (con't)

page<6:0> .....	4-127	vinten<5> .....	4-108
palet0-F<5:0> .....	4-82	vretrace<3> .....	4-143
pancomp<5> .....	4-84	vrsten<7> .....	4-123
pas<5> .....	4-120	vsyncend<3:0> .....	4-108
pas<5> .....	4-81	vsynccoeff<5> .....	4-123
pelwidth<6> .....	4-84	vsyncpol<7> .....	4-145
plwren<3:0> .....	4-149	vsyncstr<2> .....	4-98
prowsan<4:0> .....	4-99	vsyncstr<6:5> .....	4-124
rammapen<1> .....	4-144	vsyncstr<7:0> .....	4-107
rdmapsl<1:0> .....	4-136	vsyncstr<7> .....	4-98
rdmode<3> .....	4-137	vtotal<0> .....	4-98
refcol<3:0> .....	4-134	vtotal<1:0> .....	4-124
rot<2:0> .....	4-135	vtotal<5> .....	4-98
scale<2:0> .....	4-125	vtotal<7:0> .....	4-97
scroff<5> .....	4-148	wbmode<6> .....	4-116
sel5rfs<6> .....	4-108	wrmask<7:0> .....	4-141
selrowscan<1> .....	4-116	wrmode<1:0> .....	4-137
seqd<15:8> .....	4-146		
seqoddevmd<2> .....	4-151		
setrst<3:0> .....	4-132		
setrsten<3:0> .....	4-133		
shftldrt<2> .....	4-148		
shiftfour<4> .....	4-148		
slow256<5> .....	4-125		
srintmd<5> .....	4-137		
startadd<3:0> .....	4-122		
startadd<7:0> .....	4-103		
startadd<7:0> .....	4-104		
switchsns<4> .....	4-142		
syncrst<1> .....	4-147		
undrow<4:0> .....	4-111		
vblkend<7:0> .....	4-113		
vblkstr<3> .....	4-98		
vblkstr<4:3> .....	4-124		
vblkstr<5> .....	4-100		
vblkstr<7:0> .....	4-112		
vdispnd<1> .....	4-98		
vdispnd<2> .....	4-124		
vdispnd<6> .....	4-98		
vdispnd<7:0> .....	4-109		
videodis<4> .....	4-144		
vidstmx<5:4> .....	4-86		
vintclr<4> .....	4-108		

# Index

## A

aborting current drawing instruction 4-61  
accelerated graphics modes 4-124  
access to the drawing engine 5-16  
address  
    initialization 5-28  
    wrap 4-114  
address generator extensions register 4-120  
Address Processing Unit 2-2, 2-4  
addresses  
    byte 4-109  
    double word 4-109  
    single word 4-109  
alignment  
    destination/source 5-30  
alpha 4-47  
    character generation 4-148  
    mode 4-81, 4-137  
aperture location 4-11 to 4-12  
APU 2-2, 2-4  
AR registers, setting 4-42  
AR0 4-17  
AR1 4-18  
AR2 4-19  
AR3 4-20  
AR4 4-21  
AR5 4-22  
AR6 4-23  
asynchronous reset 4-145  
ATTR 4-78  
    data register 4-79  
ATTR0 to ATTRF 4-80  
ATTR10 4-81  
ATTR11 4-83  
ATTR12 4-84  
ATTR13 4-85  
ATTR14 4-86  
attribute address register 4-117 to 4-118  
attribute controller  
    index 4-78  
    register 4-78  
attribute mode control register 4-81  
attributes  
    address/data select 4-117  
    controller 4-117

Power Graphic mode 4-2, 4-16  
VGA mode 4-77

auto-init vectors 4-54  
AUTOLINE 4-17 to 4-23, 4-62, 4-68 to 4-69  
AUTOLINE\_CLOSE 4-40, 5-17  
AUTOLINE\_OPEN 4-40, 5-17  
axes  
    major/minor 5-29

## B

background color 4-24, 4-46, 5-19, 5-24,  
5-33 to 5-34, 5-39  
back-to-back transactions 4-4  
base address  
    control registers 4-11  
    EPROM 4-15  
BCOL 4-24  
begin row scan cursor 4-99  
BFCOL 4-45, 4-62  
BFIFO  
    writing 4-48  
big endian processor 4-14, 4-56  
    restriction 5-6  
    support 5-2  
BIOS enable 4-14  
BIOS EPROM  
    aperture access 4-14  
    booting 4-15  
biosen strap 4-15  
bit mask register 4-133, 4-135, 4-139  
BITBLT 4-40, 5-41  
    function 4-40  
    linear 4-41  
    operations 4-45, 4-64  
    xy 4-41  
bitblt operations 5-28  
    fast 5-28  
    supported formats with expansion 5-39  
blinking 4-81  
BLIT 4-17 to 4-18, 4-20 to 4-23, 4-50 to 4-51,  
4-62, 4-67, 4-71  
    mode selection 4-45  
    operations 4-42, 4-45 to 4-46, 4-63  
BLK 4-41

- block diagram
  - MGA-2064W 2-3
  - typical implementation 1-2
- block mode 2-4
  - defined 5-23
- blue color value 4-37
- blue increment
  - diagonal axis 4-39, 5-20
  - major axis 4-38, 5-20
  - x axis 4-38, 5-26
  - y axis 4-39, 5-26
- blue start 5-20, 5-26
- BMONOLEF 4-45
- BMONOWF 4-45
- board design, product options 4-14
- Boolean ALU 2-4
- Boolean operation
  - source/destination slice 4-43
- boot space 5-2
- booting with BIOS EPROM 4-15
- border color 4-83
- BPLAN 4-45
- Bresenham parameters 5-17
- BU24BGR/BU24RGB 4-45
- BU32BGR/BU32RGB 4-45
- Bus FIFO 4-48
  - status register 4-48
- BUYUV 4-45
- byte
  - addresses 4-109
  - panning control 4-97
  - swapping 4-14
- byte values
  - setting/resetting 4-130

## C

---

- CGA emulation 4-142
- chain four 4-149
- changing the start address 5-50
- character clock 4-146
  - division 4-114
- character fonts 4-81
- character map select register 4-148
- character mode display refresh 2-2
- circuitry
  - color expansion 2-4
  - control for external devices 2-4
  - depth generation/comparison 2-4
  - dithering 2-4

- patterning 2-4
- CLASS 4-3
- clear vertical interrupt 4-106
- clipper x
  - boundary register 4-25
  - left boundary 4-25 to 4-26
  - maximum boundary register 4-27
  - minimum boundary register 4-26
  - right boundary 4-25, 4-27
- clipper y
  - bottom boundary 4-70
  - maximum boundary register 4-70
  - top boundary 4-74
  - top boundary register 4-74
- clipping
  - disable 4-26 to 4-27, 4-70, 4-74
  - limits 5-16
  - rectangle 2-4
  - window 4-26, 4-74
- clock source selection 4-142
- clocking
  - memory address counter 4-109
  - mode register 4-146
- color
  - border 4-83
  - enable comparison 4-138
  - overscan 4-83
- color compare
  - read cycle 4-138
  - register 4-132
- 'color don't care' register 4-138
- color expansion
  - BLITs 4-46
  - circuitry 2-4
  - module 4-24
- color outputs for status port 4-84
- color palette select 4-86
- color plane enable register 4-84
- color select register 4-86
- color space converter 2-4
- compatibility mode support 4-112
- composite sync 5-50
  - enable 4-123
- configuration space 5-2
  - PCI 4-7
  - reserved locations 3-2
- configurations
  - 16 bit/pixel 4-55
  - MGA-2064W 1-2

- constant shaded trapezoids 5-21
- continuity points 5-21
- continuous address/data stepping 4-4
- control aperture 4-10 to 4-11
  - base address register 4-11
  - overlaps 4-11 to 4-12, 4-15
- control circuitry (external devices) 2-4
- control register 4-10
  - address data 4-9
  - base address 4-11
- controller
  - VGA attributes 2-2
  - VGA graphics 2-2
- conversion
  - 200 to 400 line 4-98
  - color space 2-4
- count by 2 4-114
- CPU
  - data 4-116
  - latches 4-135
  - read latch register 4-116
- CRTC 2-2, 4-87
  - currently-displayed line 5-50
  - data 4-88
  - extension register 4-119
  - horizontal timing 5-45
  - index 4-87
  - mode control register 4-112
  - protect registers 4-106
  - register 4-87
  - reset 4-114
  - vertical timing 5-46
- CRTC extension
  - data register 4-119
  - index register 4-119
- CRTC0 4-89
- CRTC1 4-90
- CRTC2 4-91
- CRTC3 4-92
- CRTC4 4-93
- CRTC5 4-94
- CRTC6 4-95
- CRTC7 4-96
- CRTC8 4-97
- CRTC9 4-98
- CRTCA 4-99
- CRTCC 4-101
- CRTCD 4-102
- CRTCE 4-103
- CRTCF 4-104
- CRTC10 4-105
- CRTC11 4-106
- CRTC12 4-107
- CRTC13 4-108
- CRTC14 4-109
- CRTC15 4-110
- CRTC16 4-111
- CRTC17 4-112
- CRTC18 4-115
- CRTC22 4-116
- CRTC24 4-117
- CRTC26 4-118
- CRTCEXT 4-119
- CRTCEXT0 4-120
- CRTCEXT1 4-121
- CRTCEXT2 4-122
- CRTCEXT3 4-123
- CRTCEXT4 4-125
- CRTCEXT5 4-126
- CTRC
  - programming 5-45
- current
  - green color value 4-34
  - red color value 4-31
  - vertical count 4-66
- cursor
  - location, high order 4-103
  - location, low order 4-104
  - off 4-99
  - on 4-99
  - skew control 4-100
  - start register 4-99 to 4-100
- CXBNDRY 4-25
- CXLEFT 4-26
- CXRIGHT 4-27

## **D**

---

- DAC status register (DACSTAT) 4-127
- data ALU 0 register 4-28
- data ALU 2 register 4-29
- data ALU 3 register 4-30
- data ALU 4 register 4-31
- data ALU 6 register 4-32
- data ALU 7 register 4-33
- data ALU 8 register 4-34
- data ALU 10 register 4-35
- data ALU 11 register 4-36
- data ALU 12 register 4-37

- data ALU 14 register 4-38
- data ALU 15 register 4-39
- data FIFO 2-4
- data format 5-36, 5-42 to 5-43
- data path, host/frame buffer 2-2
- Data Processing Unit 2-2, 2-4
- data rotate
  - count 4-133
  - register 4-133
- data select register 4-117
- data select/attributes address 4-117
- data stepping/continuous address 4-4
- data write mask 4-139
- decoding
  - circuitry, PCI bus interface 2-2
  - resource 2-2
- delay
  - horizontal retrace 4-94
  - internal pipeline 4-92, 4-94
- Depth generation/comparison circuitry 2-4
- depth lines 5-17, 5-20
- destination
  - boundary 5-28, 5-36, 5-42 to 5-43
  - current address 4-20
  - end address 4-17
  - starting address 4-18
  - y origin 4-73
- DEVCTRL 4-4 to 4-5
- device
  - control register 4-4 to 4-5
  - identification register 4-6
  - response to I/O SPACE accesses 4-4
  - response to memory accesses 4-4
  - select timing 4-5
- DEVID 4-6
- diagnostic 4-141
- direct access
  - data size 4-56
  - read cache 4-61
- direct read access to frame buffer 5-2
- display enable 4-141
  - skew control 4-92
- display refresh
  - character modes 2-2
  - VGA modes 2-2
- dithering 4-55, 4-71
  - circuitry 2-4
- division
  - character clock 4-114

- VCLK 4-124
- DMAWIN 5-35, 5-43
  - data size 4-56
- dot clock
  - rate 4-146
  - scaling factor 4-123
- double word
  - addresses 4-109
  - mode 4-109
- down scaling (not supported) 5-40
- DPU 2-2, 2-4
- DR0 4-28
- DR2 4-29
- DR3 4-30
- DR4 4-31
- DR6 4-32
- DR7 4-33
- DR8 4-34
- DR10 4-35
- DR11 4-36
- DR12 4-37
- DR14 4-38
- DR15 4-39
- drawing
  - polylines 4-22
  - rectangles 4-42
  - using depth 4-41
- drawing control register 4-40, 5-16
- drawing engine
  - access 5-16
  - starting 5-17, 5-21, 5-28, 5-35, 5-40 to 5-41, 5-43
  - status 4-65
- drawing operations
  - sequencing 2-4
  - source data 4-64
- drawing window 4-26 to 4-27, 4-74
- DWGCTL 4-40, 5-16

## **E**

---

- EEPROM 4-4
  - base address 4-15
- EEPROM write enable 4-14
- 8 bit/pixel configurations 4-60
- emulation
  - CGA 4-142
  - MDA 4-142
  - monochrome 4-81
- enable



- color plane 4-84
- line graphics character code 4-81
- RAM 4-142
- set/reset planes 4-131
- set/reset register 4-131
- vertical interrupt 4-106
- VGA palette 4-118
- end horizontal blanking 4-92, 4-94
- end horizontal retrace register 4-94
- end vertical blank register 4-111
- error
  - increment 4-19, 4-23
  - term 4-21
- explanation of programming steps 5-16
- external
  - color palette index 4-86
  - interrupt 4-53, 4-65, 5-53

## **F**

---

- fast bitblt operations (FBITBLT) 4-40, 5-28, 5-41
- FCOL 4-47
- FEAT 4-128
- feature control register 4-128
- feature input 4-140
- fetches
  - 32-bit 4-146
  - word 4-146
- FIFOSTATUS 4-48
- filled object
  - left boundary 4-50
  - x left coordinate 4-49 to 4-50
  - x left/right 5-22
  - x right coordinate 4-49, 4-51
- filled trapezoid drawing 4-17 to 4-19, 4-51
- fills
  - span line 5-21
- first pixel of screen 4-73
- flushing FIFO 4-61
- foreground 4-47
  - color 5-18 to 5-19, 5-23 to 5-24, 5-33 to 5-34, 5-37, 5-39
  - color register 4-47
- format
  - linear 4-71 to 4-72
  - little endian 5-24 to 5-25
  - Windows 5-24 to 5-25
  - xy 4-71
- frame buffer 2-2, 2-4

- aperture 4-12, 4-15
- interleave 2-4
- mapping on host bus 4-142
- MGA aperture 4-125
- non-interleave 2-4
- Power Graphic mode (interleave) 5-11
- Power Graphic mode (non-interleave) 5-11
- space 5-5
- VGA mode 5-10
- function select 4-133
- funnel count value 4-63
- funnel shifter 2-4
  - control register 4-63
- FXBNDRY 4-49
- FXLEFT 4-50
- FXRIGHT 4-51

## **G**

---

- gclk
  - divider 4-14
  - signal 4-14
- GCTL 4-129
- GCTL0 4-130
- GCTL1 4-131
- GCTL2 4-132
- GCTL3 4-133
- GCTL4 4-134
- GCTL5 4-135
- GCTL6 4-137
- GCTL7 4-138
- GCTL8 4-139
- generation of SERR interrupts 4-4
- generic device function 4-3
- global initialization registers 5-16
- glue logic 1-2
- Gouraud shading 2-4, 4-47
  - trapezoids 5-21
- graphic blink logic 4-81
- graphic clock pre-scaler 4-14
- graphic controller
  - data register 4-129
  - index register 4-129
- graphic controller shift registers
  - reloading 4-146
- graphic/alphanumeric mode 4-81
- graphics mode 4-137
  - register 4-135
  - select 4-137

graphics subsystem devices 1-2  
green increment  
    diagonal axis 4-36, 5-20  
    major axis 4-35, 5-20  
    x axis 4-35, 5-26  
    y axis 4-36, 5-26  
green start 5-20, 5-26

## H

---

hard reset 4-65  
hardware swapping  
    big endian support 3-3  
'#' (hash) mark used as a symbol 5-16  
HEADER 4-7  
high order  
    cursor location 4-103  
    start address 4-101  
horizontal  
    blanking start 4-121  
    counter 5-45  
    counter extensions register 4-121  
    display enable end register 4-90  
    pel count 4-85  
    pel panning register 4-85  
    reset enable 4-121  
    retrace delay 4-94  
    retrace select 4-114  
    retrace start 4-121  
    scaling 5-40  
    scan direction 4-62  
    sync activation 4-121  
    sync polarity 4-143  
    timing 2-2  
    total 4-89, 4-121  
horizontal video half count register 4-126  
host data transfer 4-135  
host/frame buffer data path 2-2

## I

---

I (mnemonic) 4-41  
I/O  
    accesses to VGA RAMDAC 4-4  
    address select 4-142  
    space 5-3  
ICLEAR 4-52  
IDUMP 4-23, 4-40, 4-45  
    programming 5-43  
    supported formats 5-44

triggering 5-43  
IEN 4-53  
ILOAD 4-20, 4-40  
    operations 4-45, 5-35  
    supported formats 5-38  
ILOAD\_FILTER 4-17 to 4-21, 4-23, 4-40, 5-42  
ILOAD\_SCALE 4-17 to 4-21, 4-23, 4-40, 5-42  
inactivate horizontal/vertical sync 4-114  
incomplete packet 4-57  
initialization 4-10  
    address 5-28  
    slope 5-21  
input of internal palette circuit 4-81  
input status 1 register 4-141  
input status register 4-140  
INSTS0 4-140  
INSTS1 4-141  
INTCTRL 4-8  
interface  
    PCI bus 2-2  
    to system buses 1-2  
    WRAM 4-40  
interlace  
    enable 4-120  
    mode 4-126, 5-51  
interleave 2-4, 5-49  
internal palette 4-79  
    data 4-80  
internal pipeline delay 4-92, 4-94  
interrupt 4-53  
    clear register 4-52  
    control register 4-8  
    enable register 4-53  
    level 4-8  
    line routing 4-8  
    pins 4-8  
    sources 5-53  
    vertical line 4-115  
    vertical retrace 4-140

## L

---

last access cycle to palette 4-127  
left edge  
    error term 5-22  
    minor axis increment 5-22  
left edge of the trapezoid  
    for green 4-34  
    for red 4-31  
left pixel boundary 4-50

LEN 4-54  
 length 4-54, 4-72  
 LINE 4-17, 4-19 to 4-21, 4-62 to 4-63, 4-69  
     error term 4-18  
     with RPL/RSTR attribute 4-64  
 line compare 4-82, 4-96, 4-98, 4-115, 4-122  
 line drawing 4-50 to 4-51  
     with line style 4-45  
     with no linestyle 4-42  
 line function 4-40  
 line start source address 'ssa' 4-18  
 line style 4-64  
 line width, logical 4-120  
 LINE with z 4-28 to 4-39  
 LINE without auto initialization 4-22 to 4-23  
 LINE\_CLOSE 4-40, 5-17  
 LINE\_OPEN 4-40, 5-17  
 linear  
     bitblit 4-41  
     format 4-71 to 4-72  
     source addresses 5-28  
 lines  
     autoinit, non-autoinit 5-17  
     solid 5-17  
     with depth 5-17, 5-20  
     with linestyle 5-17  
 linestyle  
     initializing 5-19  
     length 4-63, 5-19  
     lines 5-17  
     pattern 5-19  
 little endian format 5-24 to 5-25  
 little endian processor 3-2, 4-14  
 loading  
     AR0, AR2 4-68  
     AR5, AR6, XDST, YDST 4-69  
     CXRIGHT, CXLEFT 4-25  
     FXRIGHT, FXLEFT 4-49  
     SRC register 4-42, 4-58  
     YDST, LEN 4-72  
 logical line width 4-108, 4-120  
 low order  
     cursor location 4-104  
     start address 4-102

## **M**

---

MACCESS 4-55  
 Macintosh systems 5-6  
 map

    memory space 4-11 to 4-12  
 map A select 4-148  
 map B select 4-148  
 map mask register 4-147, 4-149  
 mapping  
     control aperture 3-3  
     general 3-3  
 Matrox  
     device identifier 4-6  
     manufacturer identifier 4-6  
 maximum  
     resolution 2-4  
     scan line 4-98  
 MDA emulation 4-142  
 memory  
     access register 4-55  
     banks 5-10  
     interleave 4-13  
     mode register 4-149  
     origin register 4-73  
     page register 4-125  
     pitch register 4-59  
 memory address counter 5-47  
     clocking 4-109  
     resetting 4-115  
 memory map select 4-137  
 memory space map 4-11 to 4-12  
 MGA  
     control aperture 4-4  
     direct access aperture 4-4  
     frame buffer aperture 4-125  
     frame buffer aperture address  
         register 4-12  
     general map 3-3  
     indirect access data register 4-9  
     indirect access index register 4-10  
     memory space 4-12  
     mode enable 4-124  
 MGA\_DATA 4-9  
 MGA\_INDEX 4-10  
 MGA-2064W  
     configurations 1-2  
     sections 2-2  
 MGABASE1 4-11  
 MGABASE2 4-12  
 minor axis error increment 4-19  
 minor axis increment 4-19, 4-22, 5-17  
 MISC 4-142  
 miscellaneous output register 4-142

- miscellaneous register 4-123, 4-137
- mode
  - 256-color 4-136
  - 9/8 dot 4-146
  - accelerated graphics 4-124
  - alpha 4-137
  - double word 4-109
  - graphic/alphanumeric 4-81
  - graphics 4-137
  - interlace 5-51
  - interlaced 4-126
  - MGA, enable 4-124
  - odd/even 4-135, 4-142, 4-149
  - Power Graphic 4-124
  - read 4-135
  - shift register interleave 4-135
  - Super VGA alpha 4-124
  - word/byte 4-114
  - write 4-135
- monitor
  - sync capability 4-98
- monochrome
  - emulation 4-81
  - source data 4-20
- move cursor 4-100
- multiplicator factor, linearizing ydst 4-59
- multi-purpose
  - address 0 register 4-17
  - address 1 register 4-18
  - address 3 register 4-20
  - address 5 register 4-22

## N

---

- 9/8 dot mode 4-146
- non-interleave frame buffer 2-4
- non-Super VGA display 4-3
- NOZCMP 4-41
- number of
  - displayed characters per line 4-90
  - displayed lines per frame 4-107
  - free locations, Bus FIFO 4-48
  - lines per frame 4-143
  - lines to scale 5-40 to 5-41
  - pixels per line 5-36
  - pixels/lines to draw 4-54

## O

---

- object types 5-16

- odd/even mode 4-142, 4-149
  - chain enable 4-137
  - page bit 4-142
  - select 4-135
- offset register 4-108
- offsets
  - pattern 5-24
- 128K window 4-125
- opacity/translucidity 4-44
- operating mode register 4-56
- operation code 4-40
- OPMODE 4-56
  - half-word access to 5-36, 5-43
- OPTION 4-13
- overflow register 4-96
- overscan 5-50
  - color 4-83

## P

---

- P5/P4 select 4-82
- page bit for odd/even mode 4-142
- palette
  - address source 4-79
  - entry register 4-80
  - internal 4-79
  - last access size 4-127
  - read/write examples 4-79
- parity 4-4
  - error detection 4-5
- PAT 4-58
- PAT0 4-58
- pattern 5-24, 5-31
  - address 5-31
  - offsets 5-24
  - origin 5-24
  - register 4-58
  - storage 5-24
  - x offset 4-63
  - y offset 4-63
- pattern/source address 4-20
- patterned trapezoids 5-21
- patterning 4-46, 4-71
  - circuitry 2-4
  - operations 5-28
- pattern-pixel pinning 4-58
- PCI
  - access 4-48, 4-65
  - configuration space 4-7, 4-10
  - device identifier 4-6

- interrupt line 4-8
- manufacturer identifier 4-6
- master device 4-4
- memory space start address 4-12
- read access 4-66
- read cycle 4-48, 4-65
- specification non-compliance 4-4
- specification requirement 4-3, 4-15
- PCI bus interface 2-2
  - decoding circuitry 2-2
  - specification 5-6
- pel
  - count horizontal 4-85
  - panning compatibility 4-82
  - width 4-82
- picking interrupt 4-52 to 4-53, 4-65, 5-53
- picture element
  - horizontal count 4-85
  - panning compatibility 4-82
  - width 4-82
- pins 2-3
- PITCH 4-59
- pitch 4-59, 5-16
  - of source operand 'sync' 4-22
  - range 4-71
- pixel
  - format 4-45, 5-11, 5-16
  - re-formatting 5-40
- pixel width
  - for drawing 4-55
  - panning compatibility 4-82
- planar bitblt restriction 5-34
- plane
  - masking 4-81
  - selection 5-34
  - write mask 5-16
  - write mask register 4-60
- PLL
  - control/programming 4-142
- PLNWT 4-60
- '+' sign used as a symbol 5-16
- polarity
  - horizontal sync 4-143
- polyline operations 4-67, 4-69, 4-71
- polylines 5-17
- Power Graphic mode 4-124
  - (interleave) frame buffer 5-11
  - (non-interleave) frame buffer 5-11
  - attributes 4-2, 4-16

- Power PC mode 4-14
- power up sequence 5-14
- precedence order
  - aperture 4-11 to 4-12, 4-15
- prefetchability 4-11 to 4-12
- PREP systems 5-6
- preset row scan 4-97
- processing
  - trapezoid edge 2-4
  - vector slope 2-4
- product ID bits 4-14
- programming
  - CRTC 5-45
  - explanation of steps 5-16
  - IDUMP 5-43
  - Power Graphic mode 5-49
- protect CRTC registers 4-106

## **R**

---

- RAM
  - enable 4-142
- rapid switching between sets of colors 4-86
- read map select 4-134
- read mode select 4-135
- rectangle 5-21
  - clipping 2-4
  - initialization 5-22
- red increment
  - diagonal axis 4-33, 5-20
  - major axis 4-32, 5-20
  - x axis 4-32, 5-26
  - y axis 4-33, 5-26
- red start 5-20, 5-26
- reference color 4-132
- refresh
  - counter 4-13
  - request 4-13
  - request priority 4-13
- register
  - address generator extensions 4-120
  - attribute address 4-117 to 4-118
  - attribute controller 4-78
  - attribute mode control 4-81
  - background color 4-24
  - bit mask 4-139
  - Bus FIFO status 4-48
  - character map select 4-148
  - class code 4-3
  - clipper x boundary 4-25

register (continued)

- clipper x maximum boundary 4-27
- clipper x minimum boundary 4-26
- clipper y maximum boundary 4-70
- clipper y top boundary 4-74
- clocking mode 4-146
- color compare 4-132
- 'color don't care' 4-138
- color plane enable 4-84
- color select 4-86
- CPU read latch 4-116
- CRTC 4-87
- CRTC extension 4-119
- CRTC mode control 4-112
- cursor location high 4-103
- cursor location low 4-104
- cursor start 4-99 to 4-100
- DAC status 4-127
- data ALU 0 4-28
- data ALU 2 4-29
- data ALU 3 4-30
- data ALU 4 4-31
- data ALU 6 4-32
- data ALU 7 4-33
- data ALU 8 4-34
- data ALU 10 4-35
- data ALU 11 4-36
- data ALU 12 4-37
- data ALU 14 4-38
- data ALU 15 4-39
- data rotate 4-133
- data select 4-117
- device control 4-4 to 4-5
- device identification 4-6
- drawing control 4-40
- enable set/reset 4-131
- end horizontal blanking 4-92
- end horizontal retrace 4-94
- end vertical blank 4-111
- feature control 4-128
- foreground color 4-47
- funnel shifter control 4-63
- graphic controller 4-129
- graphic mode 4-135
- header 4-7
- horizontal counter extensions 4-121
- horizontal display enable end 4-90
- horizontal pel panning 4-85
- horizontal total 4-89

register (continued)

- horizontal video half count 4-126
- input status 4-140
- input status 1 4-141
- interrupt clear 4-52
- interrupt control 4-8
- interrupt enable 4-53
- length 4-54
- line compare 4-115
- map mask 4-147
- maximum scan line 4-98
- memory access 4-55
- memory mode 4-149
- memory origin 4-73
- memory page 4-125
- memory pitch 4-59
- MGA control aperture base address 4-11
- MGA frame buffer aperture address 4-12
- MGA indirect access data 4-9
- MGA indirect access index 4-10
- miscellaneous 4-123, 4-137
- miscellaneous output 4-142
- multi-purpose address 0 4-17
- multi-purpose address 1 4-18
- multi-purpose address 3 4-20
- multi-purpose address 5 4-22
- offset 4-108
- operating mode 4-56
- option 4-13
- overflow 4-96
- overscan color 4-83
- palette entry 4-80
- pattern 4-58
- plane write mask 4-60
- preset row scan 4-97
- read map select 4-134
- reset 4-61, 4-145
- ROM base address 4-15
- sequencer 4-144
- set/reset 4-130
- sign 4-62
- source 4-64
- start address high 4-101
- start address low 4-102
- start horizontal blanking 4-91
- start horizontal retrace pulse 4-93
- start vertical blank 4-110
- status 4-65
- underline location 4-109

- register (continued)
  - vertical count 4-66
  - vertical counter extensions 4-122
  - vertical display enable end 4-107
  - vertical retrace end 4-106
  - vertical retrace start 4-105
  - vertical total 4-95
  - x address (boundary) 4-49
  - x address (left) 4-50
  - x address (right) 4-51
  - x destination address 4-67
  - xy end address 4-68
  - xy start address 4-69
  - y address 4-71
  - y destination and length 4-72
  - z-depth origin 4-75
- register space 5-3
- registers
  - Configuration Space 4-2
  - global initialization 5-16
  - Power Graphic mode 4-16
  - VGA mode 4-77
- reserved locations
  - configuration space 3-2
- reset 4-61, 5-13
  - asynchronous 4-145
  - memory address counter 4-115
  - register 4-61, 4-145
  - synchronous 4-145
- reset bit
  - minimum duration 4-61
- reset values
  - VGA mode registers 4-77
- reset/set 4-130
- resolution, maximum 2-4
- resource decoding 2-2
- response to special cycles 4-4
- restriction
  - big endian processors 5-6
  - planar bitblts 5-34
- retry cycles 4-14
- right edge
  - error term 5-22
  - major axis increment 5-22
  - minor axis increment 5-22
- right pixel boundary 4-51
- ROM
  - aperture 4-11
  - base address register 4-15

- ROMBASE 4-15
- row scan
  - counter 5-47
  - counter clock 4-98
  - cursor 4-100
  - cursor begin 4-99
- RPL 4-41
- RST 4-61
- RSTR 4-41

## S

---

- sample
  - Power Graphic mode config. space register
    - description 4-2
  - Power Graphic mode register
    - description 4-16
  - VGA mode register description 4-77
- scaling
  - horizontal 5-40
  - operations 5-40
  - supported formats 5-40
  - up 5-40
  - vertical 5-40
- screen
  - logical line width 4-108
  - off 4-146
  - origin 5-16
- sections of the MGA-2064W chip 2-2
- select
  - background intensity 4-81
  - blink enable 4-81
  - clock source 4-142
  - color palette 4-86
  - P5/P4 4-82
  - row scan counter 4-114
- selected
  - line 4-71
  - operation 4-40
- SEQ 4-144
- SEQ0 4-145
- SEQ1 4-146
- SEQ2 4-147
- SEQ3 4-148
- SEQ4 4-149
- sequencer
  - data register 4-144
  - index register 4-144
- sequencing
  - drawing operations 2-4



SERR interrupt generation 4-4  
 SERRN assertion 4-5  
 set/reset 4-130  
 SGN 4-62  
     register field setting 4-42  
 shading 4-55  
     Gouraud 2-4  
 shadow memory 4-15  
 SHIFT 4-63  
     register field setting 4-43  
 shift four 4-146  
 shift register interleave mode 4-135  
 shift/load rate 4-146  
 sign of  
     delta x (line draw/ left trap. edge) 4-62  
     delta x (right trapezoid edge) 4-62  
     delta y 4-62  
     delta y minus delta x 4-62  
 sign register 4-62  
 signal  
     DACRD/ 2-3  
     DACWT/ 2-3  
     EXTCS/ 2-3  
     EXTINT/ 2-3  
     EXTRST/ 2-3  
     GCLK 2-3  
     HIZ/ 2-3  
     MA<8:0> 2-3  
     MACK/ 2-3  
     MBE<7:0>/ 2-3  
     MCAS<1:0>/ 2-3  
     MDQ<63:0> 2-3  
     MDSF<2:0> 2-3  
     MOE<1:0>/ 2-3  
     MRAS<2:0>/ 2-3  
     MRQ/ 2-3  
     MSC 2-3  
     MSOE<1:0>/ 2-3  
     PAD<31:0> 2-3  
     PCBE<3:0>/ 2-3  
     PCLK 2-3  
     PDEVSEL/ 2-3  
     PFRAME/ 2-3  
     PIDSEL 2-3  
     PINTA/ 2-3  
     PIRDY/ 2-3  
     PPAR 2-3  
     PRST/ 2-3  
     PSTOP/ 2-3  
     PTRDY/ 2-3  
     ROMCS/ 2-3  
     VCBLNK/ 2-3  
     VCLK 2-3  
     VCLKSL<1:0> 2-3  
     VHSYNC/ 2-3  
     VIDRST 2-3  
     VLDCLK 2-3  
     VODD 2-3  
     VVSYNC/ 2-3  
 signed y displacement 4-23  
 simultaneous writes 4-147  
 single word addresses 4-109  
 16 bit/pixel configurations 4-55, 4-60  
 16 Kbyte control aperture 4-11  
 64K window 4-125  
 slope initialization 5-21  
 snooping 4-4  
 soft reset 4-61, 4-65  
     chip strapping 4-61  
 solid lines 5-17  
 source  
     current address 'sca' 4-20  
     dimension for x axis 4-19  
     linear/non-linear 5-35  
     register 4-64  
     source end address 'sea' 5-28, 5-30, 5-43  
     source start address 'ssa' 5-28, 5-30, 5-43  
     source y increment 'syinc' 5-28, 5-30, 5-43  
     source/pattern address 4-20  
 span line  
     fills 5-21  
     initialization 5-22  
 special cycle response 4-4  
 split  
     data transfer 2-4  
     screen 4-115  
 SRC register loading 4-42, 4-58  
 SRC0 4-64  
 SRC1 4-64  
 SRC2 4-64  
 SRC3 4-64  
 start  
     horizontal blanking 4-91  
     horizontal retrace pulse 4-93  
     vertical blanking 4-96, 4-98, 4-110  
 start address 4-120  
     changing 5-50  
     high order 4-101



- high register 4-101
- latch 4-98
- low order 4-102
- low register 4-102
- STATUS 4-65
- strap
  - biosen 4-15
  - state read 5-13
  - vgaboot 4-3, 4-13
- Super VGA
  - alpha modes 4-124
  - display 4-3
- support
  - compatibility mode 4-112
- supported
  - linearization pitches 4-59
  - resolutions 5-10
- switch sense 4-140
- sync capability of monitor 4-98
- sync polarity
  - vertical 4-143
- synchronous reset 4-145
- system
  - bus interfaces 1-2
  - performance 4-12
- systems
  - Macintosh 5-6
  - PREP 5-6

## T

---

- target aborting 4-5
- termination
  - current memory cycle 4-61
  - DMA sequence 4-57
- TEXTURE\_TRAP 4-31 to 4-40, 4-45
- textured trapezoids 5-21
- 32-bit
  - address space 4-11
  - fetches 4-146
- timing
  - horizontal 2-2
  - vertical 2-2
- toggle cursor on/off 4-99
- total horizontal scan period 4-89
- total source pixels 5-36
- transactions, back-to-back 4-4
- translucidity/opacity 4-44
- transparency 4-71
  - color enabling 4-46

- TRAP 4-20, 4-22 to 4-23, 4-40, 4-62
  - error term 4-21
  - operations 4-42
  - with RPL/RSTR attribute 4-64
  - with z 4-28 to 4-39
- TRAP or LINE using depth mode 4-64
- trapezoid
  - edge processing 2-4
  - left edge 4-28
  - solid line/constant 4-42
- trapezoid drawing 4-50, 4-71
  - with no patterning 4-42
- trapezoids 5-21
  - constant shaded 5-21
  - Gouraud shaded 5-21, 5-26 to 5-27
  - initialization 5-22
  - patterned 5-21
  - textured 5-21
- 24 bit/pixel configurations 4-60
- 256-color mode 4-136
- 256K memory size 4-149
- 200 to 400 line conversion 4-98
- two-operand BLIT algorithms 4-20
- typical implementation block diagram 1-2

## U-V

---

- underline location register 4-109
- underline row 4-109
- up scaling 5-40
- VCOUNT 4-66
- vector
  - 'continuity points' 5-21
  - auto-init 4-54
  - drawing 4-67
  - length 5-17
  - quadrant 5-22, 5-29
  - slope processing 2-4
  - starting x coordinate 4-68
  - starting y coordinate 4-68
  - using line style 4-46
- vertical blank
  - changing the start address 5-50
- vertical blanking
  - start 4-122
- vertical count register 4-66
- vertical counter 5-47
- vertical counter extensions register 4-122
- vertical display enable end 4-96, 4-107, 4-122

- vertical interrupt
  - clear 4-106
  - enable 4-106
- vertical line interrupt 4-52 to 4-53, 4-65, 4-115, 5-53
- vertical reset enable 4-121
- vertical retrace 4-141
  - end register 4-106
  - interrupt 4-140
  - start 4-96, 4-105, 4-122
- vertical retrace start 4-96
- vertical scaling 5-40
- vertical sync
  - activation 4-121
  - interrupts 5-53
  - polarity 4-143
- vertical timing 2-2
  - parameters 4-114
- vertical total 4-95 to 4-96, 4-122
- VGA
  - attributes address 4-118
  - attributes controller 2-2
  - display refresh 2-2
  - frame buffer 4-4
  - graphics controller 2-2
  - I/O map enable 4-13
  - index summary of registers 3-10
  - mode attributes 4-77
  - mode register reset values 4-77
  - palette enable 4-118
- VGA mode frame buffer 5-10
- vgaboot strap 4-3, 4-13
- video
  - clock 4-146
  - status multiplexer 4-84
- video memory map
  - writes 4-147
- VSYNC
  - interrupt 4-65
  - status 4-65
- vsync period 4-95

## **W**

---

- window
  - 128K 4-125
  - 64K 4-125
- Windows format 5-24 to 5-25
  - pattern 4-58
- word fetches 4-146

- word/byte mode 4-114
- WRAM
  - access type 4-41
  - interface 4-40
- WRAM-to-WRAM BITBLT operations 4-45
- wrap address 4-114
- write mode select 4-135
- write/bit function 2-4
- writing to BFIFO 4-48

## **X**

---

- x address
  - (boundary) register 4-49
  - (left) register 4-50
  - (right) register 4-51
- x coordinate of destination address 4-67
- x destination address register 4-67
- x end address 4-17, 4-22
- x start 4-22, 5-17
- XDST 4-67
- xy
  - bitblit 4-41
  - end address register 4-68
  - format 4-71 to 4-72
  - source addresses 5-28
  - start address register 4-69
- XYEND 4-68
- XYSTRT 4-69

## **Y**

---

- y address register 4-71
- y coordinate range 4-71
- y destination 4-71
  - and length register 4-72
  - value 4-72
- y end 4-19, 4-23
- y increment 4-59
- y length 5-36
- y linearization 4-59
- y start 4-23, 5-17, 5-22, 5-28, 5-36, 5-42 to 5-43
- YBOT 4-70
- YDST 4-71
- YDSTLEN 4-72
- YDSTORG 4-73
- YTOP 4-74

## Z

---

- z diagonal increment 5-20
- z drawing mode 4-41
- z increment
  - diagonal axis 4-30
  - for x 5-26 to 5-27
  - for y 5-26 to 5-27
  - major axis 4-29
  - x axis 4-29
  - y axis 4-30
- z major increment 5-20
- z origin 5-16
- z start 5-20, 5-26 to 5-27
- z value for the current pixel 4-28
- z-depth origin register 4-75
- ZE 4-41
- ZGT 4-41
- ZGTE 4-41
- ZI 4-41
- ZLT 4-41
- ZLTE 4-41
- ZNE 4-41
- ZORG 4-75





---

## *Notes*



---

## *Notes*